AFRL-SR-BL-TR-99-

0196

REPORT DOCUMENTATION PAGE

1. REPORT DATE *(DD-MM-YYYY)* 28-05-1999	2. REPORT TYPE Final	3. DATES COVERED *(From - To)* 01-08-1998 -- 28-05-1999

4. TITLE AND SUBTITLE Real Time Intelligent Coaching for Command and Control Systems: Phase 1 Final Report	5a. CONTRACT NUMBER F49620-98-C-0048
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Debra C. Evans, Ph.D.	5d. PROJECT NUMBER STTR/TS
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Micro Analysis & Design, Inc. 4900 Pearl East Circle, Suite 201E Boulder, CO 80301	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Office of Scientific Research 4040 Fairfax Drive, Suite 500 Arlington, VA 22203-1613	10. SPONSOR/MONITOR'S ACRONYM(S) AFOSR
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

19990823 142

12. DISTRIBUTION/AVAILABILITY STATEMENT
No restrictions

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

A study was undertaken to determine the feasibility of using Bayesian network technology to inform guidance supplied by an automated intelligent coach for command and control operations. To do so, and expert model, a method for comparing expert behavior and operator behavior, a Bayesian network, assessor, and feedback report were constructed. The automated tool was examined for functionality. It was determined that the approach was promising and that further extension of it is probably warranted.

15. SUBJECT TERMS
coaching, command and control, training, decision aids

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON / 19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	none	139	724-869-3411 / Debra C. Evans

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Best Available Copy

DTIC QUALITY INSPECTED 4

Real Time Intelligent Coaching for Command and Control Systems: Phase I Final Report

Table of Contents

Real Time Intelligent Coaching for Command and Control Systems: Phase I Final Report

Background

This report contains the objectives and findings of Air Force Office of Scientific Research (AFOSR) contract F49620-98-C-0048, "STTR-I Real Time Intelligent Coaching for Command and Control." This work was performed as a Small Business Technology Transfer Research (STTR) effort by Micro Analysis & Design, Inc. (MA&D), the University of Pittsburgh, and Veridian Corporation. All participants for this effort are listed in Appendix D.

MA&D, the University of Pittsburgh, and Veridian undertook to determine the feasibility of using Bayesian network technology to support automated performance assessment and intelligent coaching of personnel using a command and control (C^2) workstation. The work was designed to begin addressing the a perceived need to develop a technology that will supply, to personnel in C^2 task situations, performance feedback after high demand events.

In today's world, there are many time-critical decisions to be made by personnel in various high demand occupations. Such occupations include air traffic control, emergency response work, and military mission-critical teams, such as those operating in Combat Information Centers (CICs) on naval vessels and those directing aircraft from Airborne Warning and Control System (AWACS) platforms. Often personnel with these types of occupations find the environments in which they work to be very stressful, due to the need for quick response, the high workload level, the criticality of the decisions to be made and the cost of errors. All of these factors, in isolation and combined, can undermine the individual's or team's ability to make correct decisions by impacting the time to gather and thoroughly process information. Additionally, there may be a lack of information availability or a degradation of information content, which can have negative impact on information utility. Such situations can lead the personnel involved to make judgments that are not well-informed.

Timely and specific intelligent feedback or coaching, delivered while on the job or during training, might benefit teams or individuals by ensuring that they can produce decisions based on the best available information. This type of intelligent feedback can be of use at several points over the span of the operator's job. First, during training, such feedback could be used directly by the operator to inform him/herself about strengths and weaknesses in either overt behaviors or cognitive task performance or both. The instructor would also be able to use this information in the design of follow-on exercises for subsequent training. It would be especially useful if such intelligent feedback could drive an exercise authoring and development system such that the feedback would "pre-select" training objectives as the basis of the next exercise.

During task performance, intelligent feedback or coaching could be used at three points. First, the coach could be used before the operator begins his/her scheduled tasks to provide some initial guidance as to events to expect and actions to take in response to those events. During the operator's shift, the intelligent coach could provide guidance and information during lulls in activity. Giving feedback during lulls in the action would allow the operator time to reflect on the information, and, thus, potentially allow him/her to have better access to it during times of higher workload. Finally, at the end of the operator's shift, the intelligent coach would supply the operator with a debrief of his/her performance during the shift, with strengths and weaknesses highlighted and specific suggestions for ways to attain better performance during subsequent shifts. This debrief would focus on both decision outcomes (e.g., number of kills or safe landings) and decision processes (e.g., level of situation awareness or knowledge of heuristics). Carolan and his colleagues (1997) explored a similar approach performed for the Navy.

Additionally, such a tool would have great utility for the research community studying decision-making, as well. It would serve as a way to gather and process data during research trials or during longitudinal studies using real-world events. The focus on the measurement process variables would allow researchers to gain a greater understanding of various aspects of decision-making activities.

Finally, to be of greatest overall use, such an intelligent coach should be developed using methods that are not domain-specific. Domain independence of the methods would ensure their transportability and utility to fields other than the one in which they are initially developed. Domain independence should be supported by platform independence, as well. Thus, the methods embodied in an intelligent coach should be usable with any type of C^2 workstation, given that there is a means by which data can be accessed from the station and input into the intelligent coach, and methods whereby feedback can be delivered to the trainee by the coach.

Additionally, the impact of the intelligent coach upon performance must be quantifiable. Thus, methods for determining its utility, and that of other feedback techniques, within complex, dynamic environments must be developed. However, given the fluidity of the events that occur in such environments, often it is very difficult to determine what is a correct operator action, incorrect action, or one that has no impact until after the current action has occurred and subsequent events and actions have intervened. This means that the coach must be able to predict appropriate performance based on sparse or ambiguous performance data, and respond with information to the user. Given the sparseness of information upon which a coach must base feedback, even if it includes an expert model to help predict performance, it may not make sense to down-grade its rated performance early during a scenario or set of activities to which the operator must respond.

As mentioned earlier, we examined the feasibility of using Bayesian networks as the heart of our performance assessment process that would supply information to an intelligent coach function. Until recently, the complexity of the assessments required by trainee modeling made Bayesian reasoning computationally intractable. However, a revolutionary technology has been developed for doing complex Bayesian reasoning quickly. It is based on Bayesian Networks (also called Belief Networks, Causal Networks or Graphical Models). Bayesian Networks are now widely used in AI applications such as medical and business decision-making. Using Bayesian networks for trainee modeling is a rapidly growing area of research (Conati, et al., 1997; VanLehn, 1996), and it is at a level of maturity that it can begin to be transitioned into even more complex, time-stressed, high-demand domains such as those represented by command

and control operations. Thus, we planned to examine structuring an intelligent coach around this new, but maturing technology.

Bayesian reasoning is most useful for diagnostic assessments, such as are needed to give intelligent feedback to trainees or operators within dynamic and informationally ambiguous domains, such as the C^2 environment. An assessment is diagnostic when the goal is to determine which of multiple sub-skills, factors or competencies are responsible for the trainee's or operator's performance. To put it in cognitive terms, the goal of a diagnostic assessment is to find out *WHAT* the trainee or operator knows. There is a focus on the process, rather than the outcomes of the task. Diagnostic assessments are typically used for planning feedback or remediation. Non-diagnostic assessments, which are often used for placement or advancement, find out only *HOW MUCH* the trainee knows and not exactly what the trainee knows. Bayesian reasoning is virtually required when the assessment of a trainee or operator is underdetermined by the available performance data. A diagnostic assessment is underdetermined when there are many different ways to explain the trainee's or operator's performance. For instance, if an operator or trainee gets a problem wrong or performs a task incorrectly and there are 10 different sub-skills required for correct performance, then it is not clear which combination of the sub-skills is missing. The data underdetermine the assessment. By acquiring more performance data, the assessment can often be disambiguated. For instance, having the trainee answer 100 problems, or ensure that an operator faces similar task situations 100 times might suffice to determine which of the 210 possible combinations of sub-skills uniquely characterize this performance. However, with complex tasks, such as the ones addressed by the proposed research, the number of competencies/factors to be assessed is quite high compared to the amount of performance data available. With such complex tasks, assessments are almost always underdetermined by the data.

When an assessment is underdetermined by the data, human trainers do not simply give up, but use their experience with other trainees to guess the most likely explanation for the trainee's behavior. In a training situation, it is better to give feedback and remediation based on a guess than no feedback at all. Bayesian reasoning is simply a principled, mechanical means of making guesses based on experience when the data underdetermine the diagnosis. Bayesian reasoning is driven by PRIORS. For each competency/sub-skill/factor, a Bayesian reasoner is given a probability, called the prior probability, that a randomly drawn trainee or operator from the target population will have that sub-skill/factor/competencies. (Note: This assumes the prior probabilities of individual competencies are independent; More complicated representations than priors are necessary otherwise.) Priors can be tabulated from assessment of a set of trainees or operators obtained by using human assessors or some other "gold standard." There are also bootstrapping techniques to calculate priors using Bayesian techniques. Essentially, the priors represent the same information as that which human trainers rely upon in diagnosing trainees when the data are ambiguous, namely, the distinction between typical and rare trainee knowledge maladies. The relationship of the prior probabilities to the resulting probabilities can be seen in Figure 1.

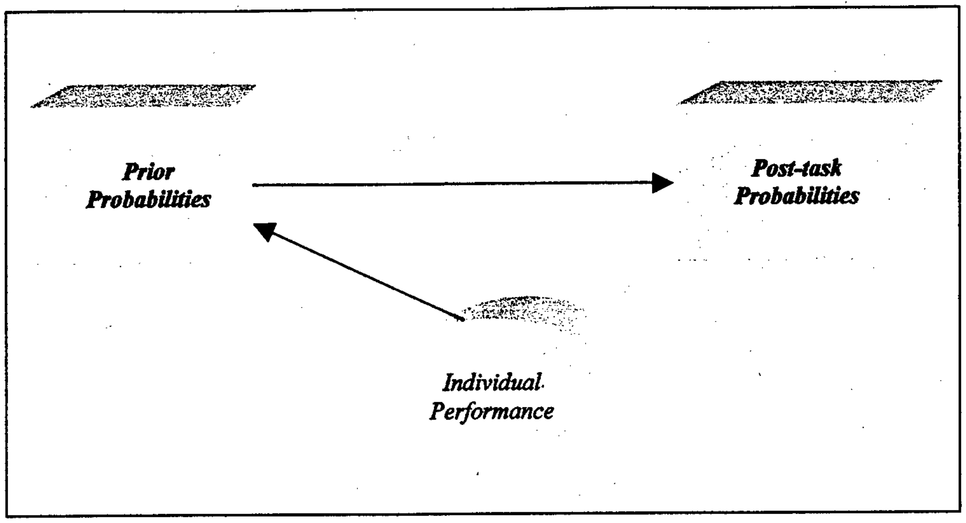


Figure 1. Relationship Between Prior Probabilities and Post-task Probabilities

The trainee modeling component of a typical intelligent tutoring system is essentially an assessment of the trainee's competence as well as their plans, goals and intentions during the performance. It is an extremely underdetermined diagnostic assessment. Bayesian reasoning should provide much more accurate diagnoses than the heuristic reasoning that is typically used. Thus, higher quality feedback should be able to be supplied to the trainee or operator.

Thus, in Phase I of this effort, our goal was to develop a proof-of-concept intelligent coaching capability consisting of four modules: 1) the expert model module (which for the proof-of-concept demonstration, contains the steps that an expert AWACS Weapons Director team would take with regard to the target task of initially committing aircraft to pursue enemy air targets), 2) the comparison module that accepts operator/team data and compares it to output from the expert model, 3) the assessment module, which takes output from the comparison module and determines, using differences in linkage weights in a Bayesian network, the possible locus and extent of performance discrepancies between the model and the operator performance, and 4) the feedback module, which places the results of the assessment module in a report, presenting information concerning competency levels. To meet our requirements we had four technical objectives:

1. Develop an intelligent coaching method, based on Bayesian network technology, to supply intelligent, well-defined, task-specific feedback to users and teams of a C^2 workstation.

2. Integrate intelligent coach output with a command and control workstation.

3. Begin to assess the impact to performance of the developed intelligent coaching method.

4. Investigate the potential of capturing verbal communications for inclusion as information sources and content into the operator model contained within the intelligent coach.

The first objective contains the primary focus of the effort, the development of an intelligent coaching method, using Bayesian network technology. The earlier discussion presents our reasons for selecting this approach for generating intelligent feedback to operators.

Our second objective was to integrate, at some level, the intelligent coach with a command and control workstation. To show that such output can be integrated with a workstation, we determined that at the present time, the best approach to integration is a "strap-on" method in which operator data would be gathered from a workstation and supplied to a processor that would generate the intelligent feedback. The feedback would then be available for review by the operators. This approach would allow the operators time to reflect upon the feedback prior to needing to implement it. It would also maintain the integrity of the operational or training system.

It should be noted that we elected to present feedback during a period of relatively low workload because of the potential negative impact that true real-time feedback can have on performance. There are two major difficulties with giving true real-time feedback to an operator (real-time feedback referring to the potential of feedback occurring after any or every action taken by an operator during task performance). First, we examined several high-demand command and control systems in which time-critical decisions must be made (for example AEGIS and AWACS). Our examination included observations of performance on such systems and discussions with subject matter experts. From this initial examination, we determined that interrupting the operator while he or she is performing the tasks at hand in order to supply feedback may result in a detriment in performance due to the intrusive nature of the feedback. This suggested that feedback should be prepared in real time, but its presentation saved until the operator can reflect upon it.

Second, in order to give feedback to an operator while in the midst of performing tasks during a critical real-time operation, the assessment and coaching system initially must predict the operator's knowledge state based on extremely limited information. In a non-training situation, the coach must also predict the knowledge states and potential behaviors of others in the operation whose knowledge states and behavior patterns may be unknowable, for example, pilots in hostile aircraft. During the early portions of the operation, the coach may not have enough information as to the state of events and availability of information to the operator to generate useful or focused feedback. In this type of situation, what may be more useful to the operator is a workstation interface that highlights the information that is the most critical for optimal task performance. However, this is a re-designed interface, not an intelligent coach. A prototype of such a system has recently been developed to support some AEGIS functions (Morrison, et. al., 1997).

Our third objective addresses the need to begin early assessment of the utility of the output of the technology. In order to determine whether a method or approach should be considered for further study, knowing its feasibility is not enough. The method, in addition to being feasible, must be useful. To meet our third objective, we intended to begin developing strategies for determining the utility of our method for using Bayesian networks for identifying areas in which operators need feedback.

Finally, our fourth objective was to examine the potential for capturing and including verbal communications into the performance analysis. Verbal communications, for high-demand, decision-making teams, is often an important method by which information is conveyed to other team members, workload is re-distributed, decision processes occur, and team cohesiveness is maintained. Frequently, capture of communication initiation and reception can occur and be correlated to other capturable environmental events available from the workstations, but often the uncapturable content of the those communications reflect details of operator knowledge that cannot be captured in any other way. Therefore, in order for any coaching or feedback mechanism to be truly reflective of the operator's knowledge state, the content of verbal communications needs to be factored into the current state of the coach's model of the operator.

The following section presents the methods we employed to address each of the identified technical objectives.

Methods

To meet our first objective, of developing a prototype automated tool that could supply intelligent feedback for performance on command and control system, we performed four major tasks. These tasks were:

- Develop an architecture for an automated tool
- Develop a prototype version of the tool
- Test the tool
- Determine the feasibility of incorporating automatic speech recognition into the tool

First, we developed a software architecture for the tool. The components can be seen in Figure 2. The components consisted of:

- a data filter
- an expert model
- a comparator
- a Bayesian network and assessor
- an output report
- an interface.

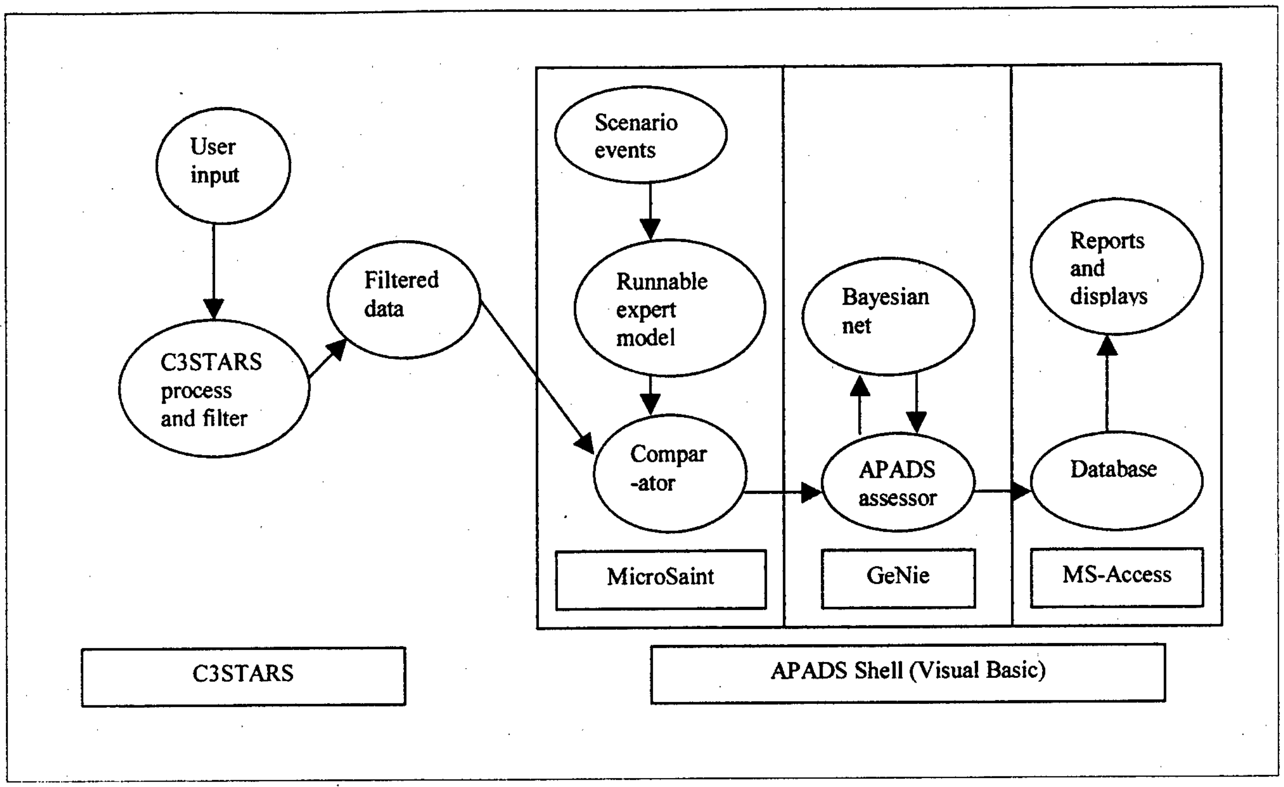


Figure 2. Relationships among APAD Components

From the figure, one can see that operators would generate behavioral data in response to a situation. They would respond to the situation using a command and control system. The operator performance data would then be fed to a comparator that would compare the operator performance to the performance of an expert model responding to the same events. The results of the comparison would be submitted to a Bayesian network that reflects the heuristics, decision types, and cognitive competencies associated with the tasks that need to be performed by the operator in response to the situation. The prior probabilities existing in the Bayesian network were modified based on the performance differences between the model's performance and the actual operator data. The new probabilities were to be fed to a database and then placed into a report that could be used to supply feedback to the operator(s).

Development of each component is discussed in the following paragraphs.

Expert Model

To develop the expert mode, we first selected a command and control task that would be appropriate to the test bed that we were using. As our testbed command and control workstation, we used the C^3 Systems Training Assessment Research Simulator (C^3STARS) at Brooks AFB. This facility is managed by the Air Force Research Laboratory Human Effectiveness Warfighter

Training Research Directorate (AFRL/HEAB) (now AFRL/HRM), from whom we have received permission for its use and access to supporting materials and research data. This is a high-fidelity simulator used for research and training of AWACS weapons directors (WDs). The simulator is configured to support up to four WDs. There are also workstations for personnel who supply the WDs with aircraft pilot input during an exercise. The C^3STARS has a few minor discrepancies from actual AWACS WD workstations. However, the simulator allows for extensive data capture, performance analysis, and scenario control. These features ensured that we would have an adequate testbed to supply us with appropriate data for development of our strategy for intelligent coaching and a location for assessing the developed coaching mechanisms.

The task focus for our effort was on complex information gathering, utilization, and dissemination in a team environment. To briefly describe the command and control tasks performed by WDs: WDs work in teams of three to six individuals, depending on the demand characteristics of the operation. They are responsible for managing air resources during air operations. WDs call up air assets, direct them with regard to ways to accomplish the mission, schedule them for re-fueling, etc. They make tactical decisions as individuals and as a team (Elliott, et al., 1997). Therefore, WDs are exemplary of command and control teams.

For our initial feasibility study, we selected the task of the initial committing of a High Value Air Asset (HVAA) to an enemy air target. To determine the cognitive steps and heuristics employed by an expert WD toward solving this problem, we interviewed a WD domain expert, Mr. Mathieu Dalrymple, of Veridian Corp. From our discussions with Mr. Dalrymple, we generated an expert model (seen in Appendix A) to depict the heuristics he described to us.

We implemented the expert model in MicroSaint, a commercial off–the-shelf tool developed by MA&D. To test the model, we re-coded a portion of an existing C^3STARS training and research scenario so that it could be included as events in MicroSaint to which the expert model could respond. The model was modified until it responded to the events in the way a real expert would.

Operator Data Filter

The C^3STARS captures operator performance data, as operators respond to events presented to them on their workstations. The events are inserted into the workstation via scenario files. The events appearing in the scenarios occur on a timeline, with the operators controlling the performance of their own assets via key press and verbal commands to researchers playing the roles of pilots, and who are using other workstations.

The simulator captures all key presses performed by both the operators and the simulated pilots. However, the simulator output filter routines can be modified to selectively extract desired types of data. Veridian Corp. developed modifications to the output filter routines such that only commit and air asset vectoring information was produced. These data were in ASCII format. An additional filter was developed to select only operator data that contained the first commit actions for friendly HVAAs. Thus, simulated pilot data and data reflecting HVAA commits subsequent to the first one were removed from the data to be compared to the expert model's behavior. Additionally, a routine was developed to reformat the data so that it can be compared to the expert model output. A routine for pasting the filtered operator data into a table within the database application that serves to connect the pieces of the tool was developed.

Comparator

The comparator component of the tool resides within MicroSaint as an additional function that is performed after the expert model completes its processing of the events to which it must respond. The model was developed so that its output is saved in an ASCII file. Then, using the programming language available in MicroSaint, a task node was developed that uses the operator data, as an event stream and compares it to the expert model output and indicates which expert and operator outcomes match or do not match. The compared data are committed HVAAs, their targets, and the vectors to be taken to the targets. The results of the task are written to an ASCII-formatted results file.

Bayesian Network and Assessor

Dr. Kurt VanLehn and his associate, Zhendong Niu, both of the University of Pittsburgh, developed the Bayesian network and assessor. To develop the Bayesian network, shown in Figure 3, the tasks in the expert model and competencies associated with those tasks were examined. (The competencies were developed through discussions with Mr. Dalrymple). The relationships among overt, measurable behaviors, cognitive tasks, and competencies were graphed. Each node in the network could be in either of two states (on/off or yes/no), and a prior probability was assigned to each node for each state. Initially all prior probabilities were set at .5. However, with more input from a domain expert as to the greater or less likelihood of any state in a node being greater than the other state, the prior probabilities can be so adjusted.

The assessor was designed to accept the results of the expert-operator comparisons on observable measures.and then modify the probabilities within the network based on the differences. For example, if the operator did not perform an observable task that the expert would have, the assessor changed the "task observed" node to zero and the "task not observed" node to 1.00. This value would then impact all of the nodes related to it, changing their values. Over repeated tasks, the probabilities for the observable tasks are modified based on the number of same or different performances of the operator versus the expert.

Finally, the assessor generates an ASCII file containing the node name and the final probability associated with the node. MA&D developed Visual Basic code to paste this file into an MS-ACCESS database table. For further details, the C++ code for the assessor may be found in Appendix B.

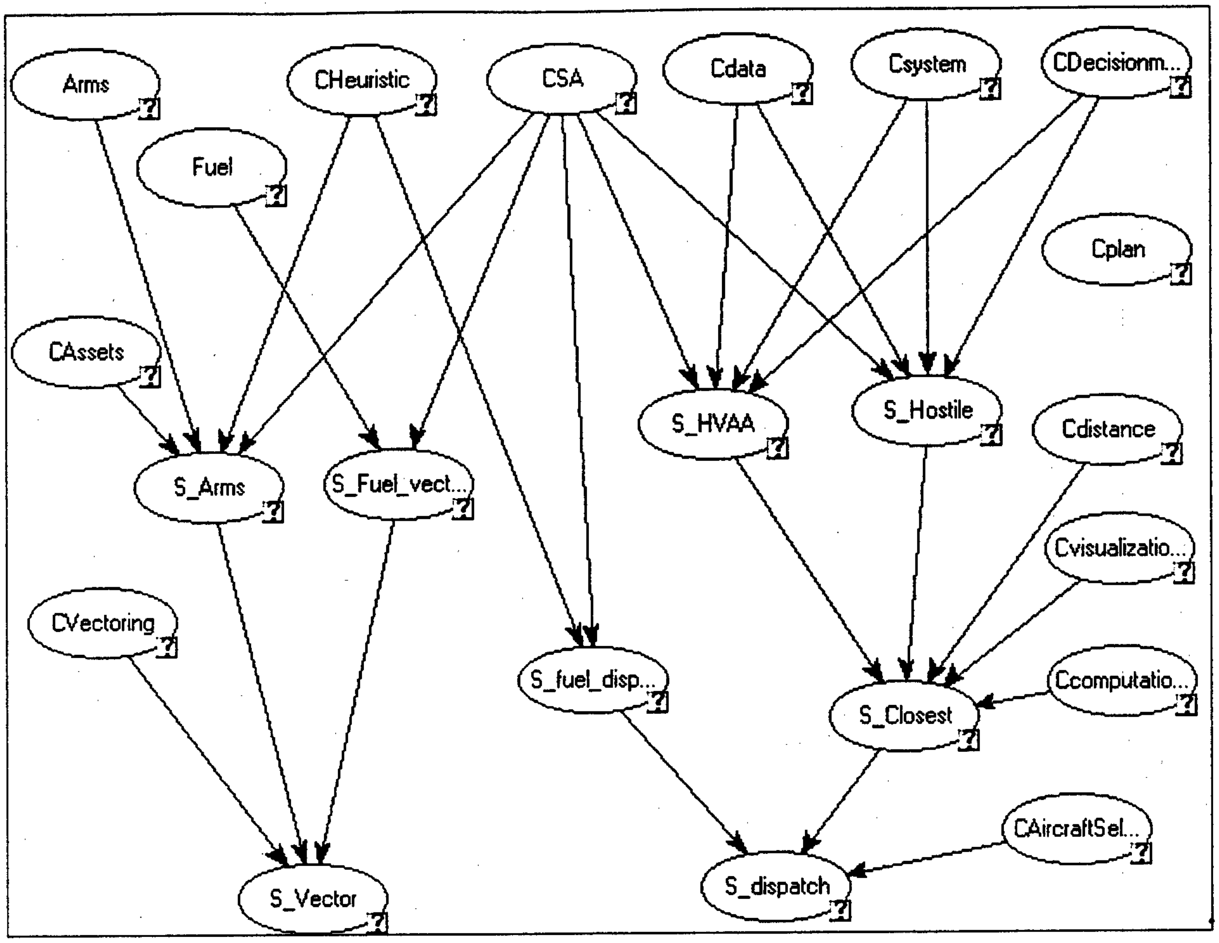
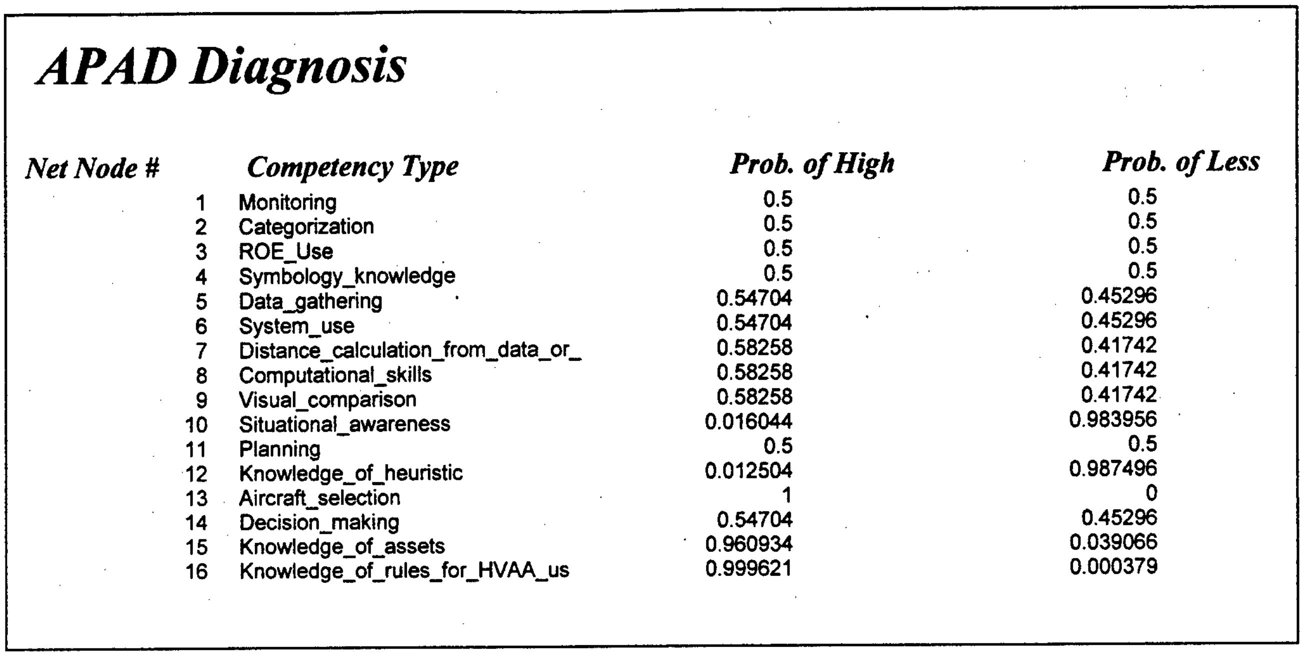


Figure 3. APAD Bayesian Net

Feedback Report

MA&D designed an MS-ACCESS report to display the results of the assessor. This report presents the node name and final probability associated with it, of all the competency and hueristics-related nodes. This report was designed to supply operators, instructors, and senior personnel with information concerning strong and weak areas of competency. This information is also of value to researchers developing process measures of performance, because the results allow them to have a measure of implicit behaviors. An example of this report is shown in Figure 4. The report displays the Bayesian network node number, the name of the competency type, the probability that the operator/team shows a high level of competency, and the probability that the operator/team show a low level of competency.



APAD Diagnosis

Net Node #	*Competency Type*	*Prob. of High*	*Prob. of Less*
1	Monitoring	0.5	0.5
2	Categorization	0.5	0.5
3	ROE_Use	0.5	0.5
4	Symbology_knowledge	0.5	0.5
5	Data_gathering	0.54704	0.45296
6	System_use	0.54704	0.45296
7	Distance_calculation_from_data_or_	0.58258	0.41742
8	Computational_skills	0.58258	0.41742
9	Visual_comparison	0.58258	0.41742
10	Situational_awareness	0.016044	0.983956
11	Planning	0.5	0.5
12	Knowledge_of_heuristic	0.012504	0.987496
13	Aircraft_selection	1	0
14	Decision_making	0.54704	0.45296
15	Knowledge_of_assets	0.960934	0.039066
16	Knowledge_of_rules_for_HVAA_us	0.999621	0.000379

Figure 4. APAD Bayesian Results Report

Interface

MA&D developed the interface for the tool, named the AWACS Performance Assessment and Diagnostic (APAD) system in Visual Basic (VB). The opening screen for the tool may be seen in Figure 5. The interface was developed to be easy to use and to require a small amount of data entry. Some of the APAD interface screens may be seen in Figures 6 and 7. The VB code also served as the integration product for the other components, calling the commercial-off-the-shelf (COTS) products that were used to build the tool.

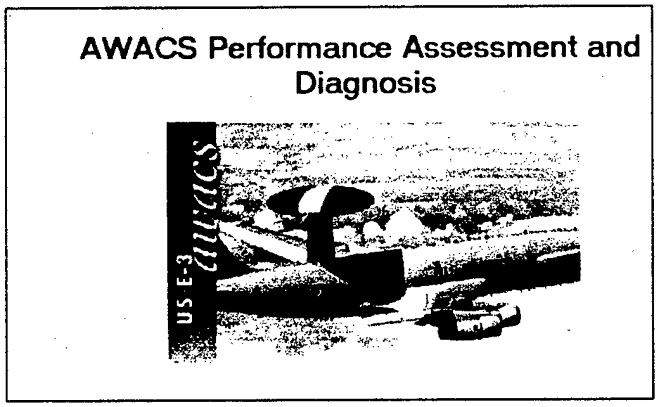
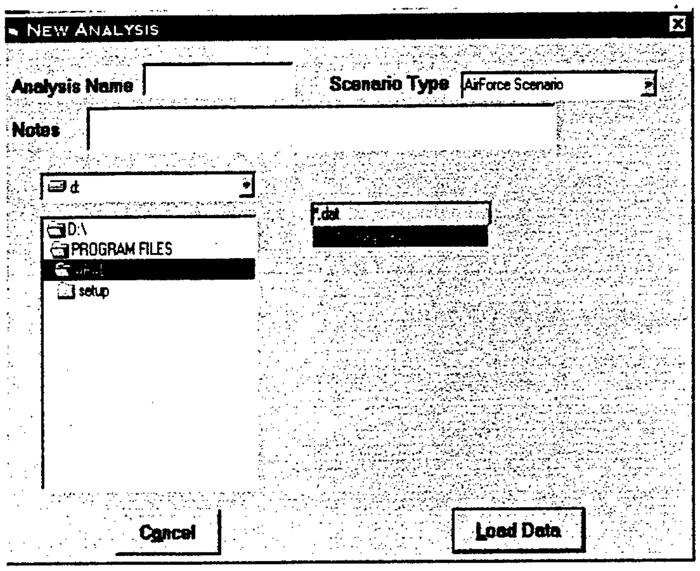


Figure 5. APAD Opening Screen



Analysis Name

Scenario Type AirForce Scenario

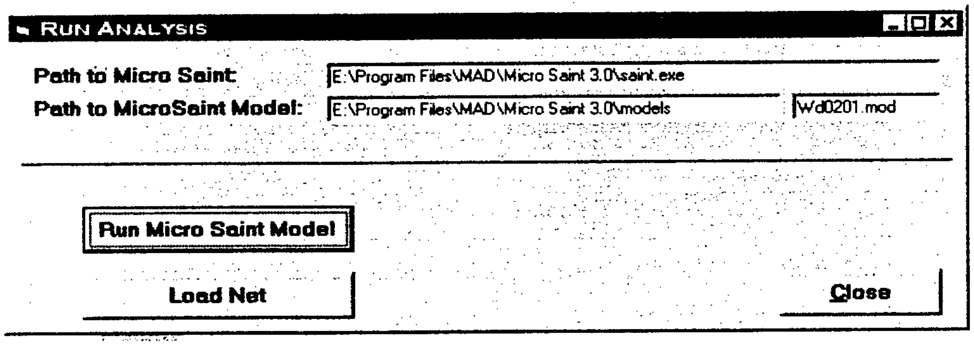
Notes

d:

D:\
PROGRAM FILES
setup

*.dat

Figure 6. APAD New Analysis Operator Data File Entry Screen



Path to Micro Saint: E:\Program Files\MAD\Micro Saint 3.0\saint.exe

Path to MicroSaint Model: E:\Program Files\MAD\Micro Saint 3.0\models Wd0201.mod

Figure 7. APAD Analysis Screen

Automatic Speech Recognition

Due to the highly communicative nature of many WD tasks, we undertook to examine the feasibility of integrating automatic speech recognition (ASR) as a way to capture a greater amount of performance data. The examination of ASR technology as it could be applied to automatic performance assessment and diagnosis resulted in a draft working paper, which appears in Appendix C.

Results

APAD Testing

APAD was constructed as described above. Having it process two levels of operator data tested its functional capabilities. The first set was constructed to by modifying the output of the expert model's performance in relation to the test scenario. These data were used to ensure that the comparison process and the assessor were functioning properly and that reasonable and understandable output was being generated.

The second test data set was derived from actual operator data. These data were from an actual C^3STARS run on a longer version of the scenario that we were using. The data contained operator (both friendly and hostile) HVAA commits, the targeted aircraft, and the vectoring information. The data file was reduced in length, since it reflected a longer version of the scenario than what was to be processed by the model. Additionally, the aircraft designators appearing in the file were double-checked to ensure that only aircraft that were in the reduced version of the scenario appeared in the data file.

The second data set was used to test the complete functionality of APAD. The interface required the user to indicate the path to the data file. The data file was accessed and parsed by the tool. Then the user indicated the location of the model and of MicroSaint. The tool opened MicroSaint and placed the model and the operator data into it. The user then ran the model to create the results and comparison files to be used by the assessor. Once the model had finished running, the user needed to close the application, before continuing. Then the user selected "Run Bayesian Analysis." At this point, the tool accessed the network and the assessor code. The assessor entered the information in the comparison file into the Bayesian network and recorded the final probabilities. The VB interface pasted the assessor results file into an MS-ACCESS database table. The user was able to call up the report of the Bayesian results. The results were then examined for their plausibility, given the contents of the operator data file.

The Bayesian output for the second data set indicated that there was a weakness in the model with regard to its reflection of the type of data presented to it. It should be noted that the model only addressed initial commit actions. However, the data included both initial and subsequent commit actions for the same HVAAs. Since the expert model did not take into account aircraft location and availability after initial commits, anytime an operator committed a previously-committed HVAA, because of its location, to a hostile air contact, there would be a flagged discrepancy between the model and the operator data. Thus the analysis was not totally accurate because of the actual spatial relationships among aircraft when the operators committed them. The resulting inaccuracy does not point to a design or feasibility problem, rather just a scope of task coverage issue.

Conclusions

Feasibility of Use of Bayesian Technology for the Development of Real Time Intelligent Coaching

After evaluating the output of APAD and demonstrating its capabilities to personnel at ARL/HRM, we concluded that a Bayesian approach to automatic performance assessment and diagnosis is feasible, if we ensure fairly complete coverage of tasks. Greater representation of the tasks to be performed, including subsequent commits, would alleviate problems that are associated with restricted task coverage.

Additionally, we determined that our architecture would not be transferable to operational systems in which the environmental events are not known prior to actual task performance. Thus, in Phase II of this effort, we intend to develop a filter, in Visual Basic, that will take a workstation's log or environment file (which it captures during operation), select the environmental events of importance, and paste them into the events queue of the MicroSaint model. In that way, the model will be able to run against events from an operational system—the same events responded to by the operator of the workstation. This would also alleviate the need for the expert model to calculate the positions of the aircraft—the environment file would supply these data.

One aspect of tool development that we have determined that requires modification is the procedure by which the Bayesian network is generated. For the effort presented here, it was produced by hand after examination of the expert model and the paths down which the expert's solution could traverse, given different environmental events. In subsequent work, we plan to develop a Bayesian network generator which will take as its input an expert model and all of its potential solutions and will output a developed network.

Finally, we realize the need for greater testing of the tool's usability and extension of the feedback methods. We can predict that such testing will indicate the need to develop stronger, more concrete and directive feedback and guidance in order to ensure that operators, supervisors, and training personnel will be able to make functional use of the information supplied by the tool. We intend to gather input from the potential user community to determine the format and level of guidance most appropriate to the proposed uses of APAD.

Our final assessment is that the approach that we selected for generating intelligent information to be used for intelligent coaching is promising. However, the information generated by the Bayesian analysis needs to tied to concrete remediation and guidance that can be used by operators, supervisors, or training personnel. Additionally, we did not strive to supply "real time coaching," in the sense that the coaching occurred during job/task performance. During task performance, coaching may not be what operators need; remedial guidance, as implied the term "coaching" may detract from operational performance. Thus, we argued that such guidance should be supplied subsequent to the operator's completion of the task set or mission. However, during a mission or task set performance, operators may benefit from intelligent aids that support their decision processes, such as highlighting critical information.

ASR Feasibility

After examining the current state of ASR technology, we cautiously concluded that this technology may have utility within a performance assessment tool. We feel that much of the ASR technology is close to the state where it can be integrated into team performance and communications assessment tools. The important word here is "integrated." There is ASR technology available that will meet many of the requirements for use in a tool such as APAD. However, work must be undertaken to bring these pieces together.

For example, the state of the technology is such that multiple speaker recognition is currently available in conference transcription software, and on-going research shows great promise for this technology. Additionally, work in environmental noise removal, or reduction, seems to be moving swiftly such that the capability should be available in an ASR tool in the very near future. ASR technology already supports continuous speech recognition and relatively easy methods for training voice files, with profiles being updated after every session with the user. Finally, there are programmer tool kits to allow ASR technology to be integrated into other applications. (Possibly, such an application kit would allow for the integration of ASR technology with a time-stamping capability needed to synchronize other captured performance data with the verbal acts.) These pieces need to be brought together in a single package before one can begin to integrate it with other components for comprehensive automated performance assessment, job aiding, or training tools.

Thus, our overall recommendation is to introduce such a capability into APAD, or a similar tool, to better see how much additional information it supplies to the diagnostic process. If it adds substantially to the process, as we think it will, then the capability for voice data capture and analysis should be fully integrated into the tool.

Future Work

In Phase I of this effort, we demonstrated the feasibility of using Bayesian networks to generate feedback for post-event reflection. To extend this work and to meet an overall objective of a stand-alone automatic debriefing tool/coach, which WD personnel can use subsequent to a mission phase or training exercise, we would need to meet six objectives.

1. Increase task coverage of the APAD tool.

2. Use environment files to supply event data.

3. Integrate automatic speech recognition (ASR) technology to augment operator data files.

4. Incorporate visual playback capabilities and summary performance statistics.

5. Extend and expand the capabilities of the Bayesian network.

6. Determine the measures of performance for such tools in order to determine their actual effectiveness.

Each of these objectives are discussed below.

Increase task coverage of the APAD tool. The current version of APAD is a prototype, designed to prove the concept that Bayesian networks can be used to generate intelligent feedback for C^2 systems. Since the tool was a feasibility demonstration, it incorporated only one domain task. Now that the feasibility of the concept has been shown, the tool can be extended to cover a wide range of mission critical WD tasks. Extending the task base will allow us to move the tool from a concept stage to a product for potential fielding. To that end, we will expand the expert model and the Bayesian network to cover all mission-critical WD tasks, such as aircraft re-fueling and tanker control tasks, bomber control, and initial track control. We will also incorporate tasks that are based on, or augmented by, verbal communications. Since many tasks performed by the WDs require verbal communications, any model of their performance that does not include verbal tasks cannot accurately portray WD mission-critical behavior. Inclusion of verbal tasks will allow us to address *all* of the important WD tasks, not a small subset. Additionally, since C^2 environments frequently include verbal performance by operators, if an APAD-like tool is to be fielded into other situations, the ability to include verbal tasks in the model is of paramount importance.

Additionally, we will develop our model so that it supports the expanded capabilities of the Bayesian network that we plan to undertake. To do so, we will build the model so that does not take actions, per se, but only recommend them. Moreover, it will be non-deterministic and buggy, in that if multiple derivable actions can be derived via correct or incorrect reasoning, it will recommend them all. The model will record the reasoning associated with recommended actions, and these will be converted into a Bayesian network.

Use environment files to supply event data. In the current version of APAD, the environment to which operators responded was specified by training scenario files that were used to stimulate the expert model and to which actual operators could respond within the C^3STARS milieu. Since the C^3STARS is a training and research device rather than an operational AWACS, its operation is stimulated by scenario event files and simulated enemy pilot responses to system users. Operational AWACS, on the other hand, functions in response to an actual detected environment. In order to be useful within an operational environment, APAD must be able to gather environment data from a functioning AWACS during a mission or portion of a mission. Then APAD must use these data to stimulate the expert model, so its responses may be compared to that of the operational team. Figure 8 displays the data flow from the C^3STARS to a revised APAD (R-APAD).

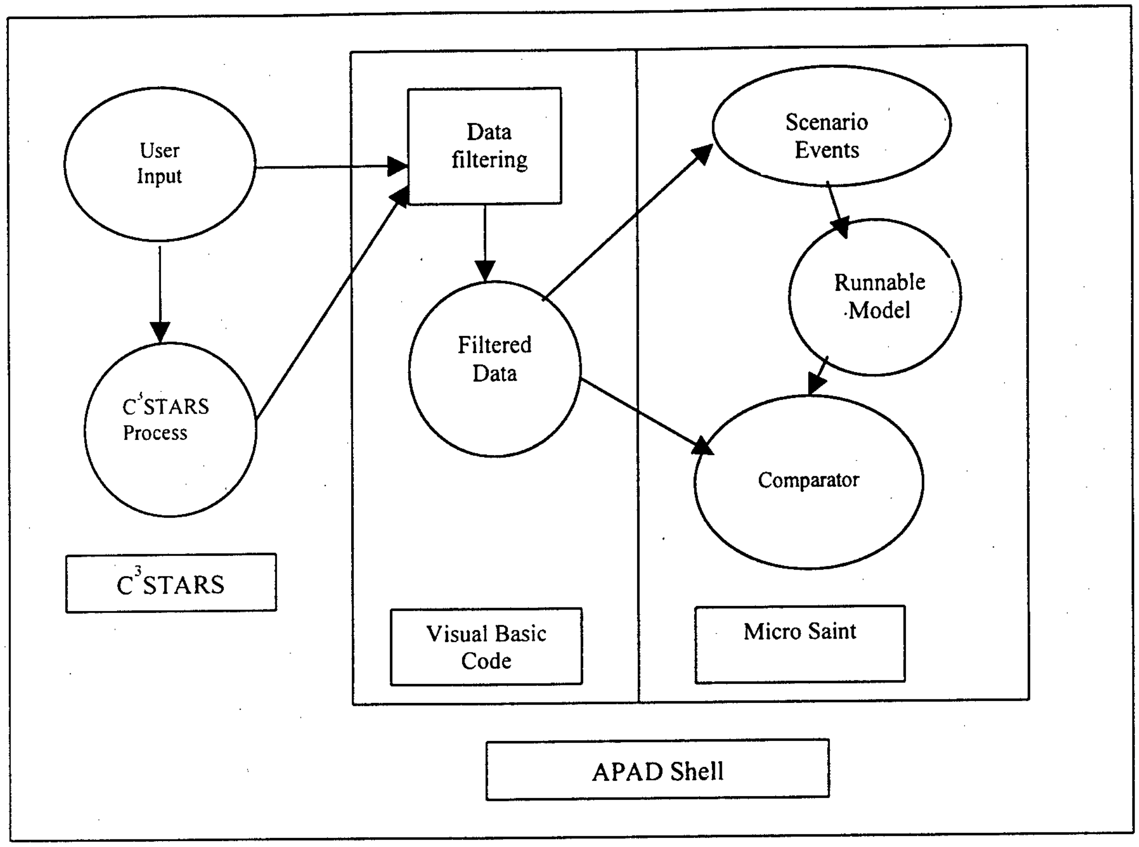


Figure 8. Data Flow from C^3STARS to R-APAD

To meet this objective, we will use environment snapshot files retrieved from the C^3STARS. These same types of files are generated and saved by an AWACS as history or log files during its operation. These files contain a snapshot of the system state; these snapshots are collected every three seconds during system operation. To use these files, we will build a filter to selectively extract the information that we need to supply the model. We will also build an interface to the model in order to supply it with the necessary information. The model will then be able to respond as it would in an operational setting.

Integrate automatic speech recognition (ASR) technology to augment operator data files. To be able to compare the performance of the model on speech tasks with the operators' performance, we need to be able to collect and analyze operator speech data. To accomplish this, we will program an ASR interface that will capture WD utterances, time-stamp them, and place them into a file that can be interleaved with the operator data file currently generated by C^3STARS.

We will also include, if possible, the capability to use APAD to supply correct, digitized versions of speech data, as part of APAD's use as a de-brief tool. The aim is to supply the operator with a correct speech model for later use.

Incorporate visual playback capabilities and summary performance statistics. One important method of supplying feedback that we have found through our research on command and control tasks is that of visual playback of the important events to which verbal feedback is referring. In that way, the operator can mentally place him or herself back into the situation to perform a self-assessment in conjunction with the automated feedback. It is similar to re-living the events to pinpoint the exact situation in which alternative decisions may have been more appropriate.

To meet this objective, we will program into the user interface the ability to present on a scope-like display the positions of critical tracks, and other information, in relation to time within the mission. We will also make available summary statistics, such as number of successful commits, kills, etc. to give the operators a full picture of their performance. Frequently, these are the types of data that operators find to be important in their assessment of themselves. These outcome data are items that are typically reported during assessment.

Additionally, we will tie the output from the Bayesian assessment more closely to guidance for performance change. During Phase I, we supplied a report concerning competency levels, based on the outcome of the Bayesian analysis. In subsequent work, we intend to tie those outcomes to constructive guidance.

Finally, we need to develop a user interface that will allow operators the ability to receive feedback subsequent to mission or training performance. This interface should be easy-to-use, and the product will require very little training to use.

Extend and expand the capabilities of the Bayesian network. The Bayesian network described in this report was developed specifically for our target task. We will need to expand the network to include new tasks. During Phase I, the network was built by hand. As mentioned earlier, we intend to develop a method to automatically generate a complex and extensive Bayesian network, using the expert model. To address this issue, we will build an expert model that will not take actions, but only recommend them. It records its reasoning, which we convert to a Bayesian net. The links lead from input events in the history file (e.g., observing a potentially hostile air contact cross into friendly air space) to state variables (e.g., the air contact is presumed hostile) to actions (e.g., directing a friendly squad to intercept it).

For each real action in the history file, there may be hundreds of actions predicted by the model. The larger variety of potential actions occurs because there is genuine ambiguity in the task (e.g., if two squads are tied for closest to the hostile, either can be committed to an intercept), there is uncertainty (e.g., it's not clear how much fuel a friendly has), and there are errors (e.g., the operator is confused about the weapons on board the friendly chosen for committing). Our goal will be to have so many actions predicted by the model that there is rarely an action in the history file that is not predicted.

When processing the operator and environment files, predicted actions will be identified as Early, On-time, Late or Never. Most actions will be identified "Never." The nodes corresponding to input events are not scheduled for update action, but instead are given a prior probability corresponding to how likely the operator is to have observed that information given the current settings on the console and the task load.

What will make this approach feasible for automatic construction is that we will assume that all tasks are instances of generic tasks, such as intercepts and walking the clock. Thus, the networks can either be built in advance in generic form and adapted to the specific situation in the history file (e.g., connecting it to a node representing the AC's time in the air, a node representing the currents ROE, etc.).

The resulting network will span the whole history file. Potentially, there will be thousands of input events and tens of thousands of predicted actions (of which hundreds are observed actions). Since this will result in a very large network, it will be updated in piece-meal fashion, but the overall outcome will be a network that is very sensitive to real and expected operator actions.

Determine the measures of performance for such tools in order to determine their actual effectiveness. Although research and commonsense suggest that supplying intelligent feedback to personnel will improve their subsequent performance, we need to establish this as a fact. To do this, we will develop methods to assess change in performance subsequent to receipt of feedback.

Initially, we will use methods that have been used in other training tool assessments. In general, we will compare the performance of WD teams who do not receive feedback to those that do receive feedback both prior to, and subsequent to, the feedback. All teams will function within the same operational or scenario environment. However, if this approach does not meet our requirements, due to the nature of the C^2 environment, we will investigate other assessment methods, such as surveys addressing the perceived utility of the feedback or analyses of process change. (It is very possible that for journeymen level personnel, the fine-tuning of performance that results from intelligent feedback may be too subtle to detect using standard product measures of performance change. In this case, we will need to determine appropriate measures of process change.)

Once a stand-alone tool has been developed, we can begin to further extend the product. There are several paths to product commercialization that may be taken. First, APAD could be included as an integral component of the AWACS. Second, the APAD architecture could be expanded to encompass other C^2 domains such as the AEGIS combat information center (CIC), air traffic control and military or rescue forces coordination.

Finally, the APAD architecture and process could become the basis for the development of an authoring tool for easy development of future automated evaluation and diagnostic products. We foresee the need for different tools to support development of intelligent coaches for command and control.

One authoring tool would be specific to training situations, in which scenario ground truth can be determined prior to use. To give this tool its greatest utility, it should be designed to integrate the development of the assessment tool and training scenarios. This means developing interfaces to scenario development modules for training workstations. One can conceive of an interface that would allow the individuals building scenarios to build as much of the expert models as possible concurrent with exercise development. As the developer identifies expected (or expert) responses and options to scenario events, information sources, etc., these same data would be supplied to a module that would build the intelligent coach. This interface for constructing models would be supported by a final authoring module that would support easily constructing the Bayesian networks necessary for the assessment module.

The second authoring tool would share components with the first. However, it would differ in that it would be used to set up general structures for capturing environmental events, as they happen, not before the fact, as in the tool for training development. As with the authoring tool for training workstations, however, this tool would allow a domain expert to input the structures for the expert models and the information needed to automatically build the components of the assessment module.

The steps presented above would result in a final product or set of products that would offer managers and trainers within various command and control environments a way to develop a customized intelligent coach and integrate it with their work or training stations. The customized coach would then collect and analyze performance data, and give appropriate feedback to the users during a debriefing session. The users would be able to use this well-tuned feedback to modify their subsequent behavior, thereby resulting in better performance.

References

Carolan, T., Evans, D. C., Roth, J. T., & Scott-Nash, S. (1997). Automating performance assessment and diagnosis in a complex domain. Paper presented at 1997 Meeting of the Human Factors and Ergonomics Society.

Conati, C., Gertner, A., VanLehn, K., & Druzdzel, M. (1997). On-line student modeling for coached problem solving using Bayesian networks. In A. Jameson, C. Paris, & C. Tasso (Eds.), User Modeling: Proceedings of the Sixth International Conference, UM97 . New York: Spring Wien.

Elliott, L. R., Dalrymple, M. A., & Neville, K. (1997). Assessing performance of AWACS command and control teams. Paper presented at 1997 Meeting of the Human Factors and Ergonomics Society.

Morrison, J. G., Kelly, R.T., Moore, R. A., & Hutchins, S. G. (1997).. Tactical Decision Making Under Stress (TADMUS) Decision Support System. Paper presented at 1997 Meeting of the Human Factors and Ergonomics Society.

VanLehn, K. (1996). Conceptual and meta learning during coached problem solving. In C. Frasson, G. Gauthier, & A. Lesgold (Eds.), ITS96: Proceedings of the Third International Conference on Intelligent Tutoring Systems. New York: Springer-Verlag.

Appendix A: Expert Model

The following pages contain:

- A visual representation of the top level APAD expert model
- A visual representation of the "Transitional Mission" level of the APAD expert model, because the scenario that was used in APAD was a transitional one; these graphics represent a left to right progression
- The complete code file for the APAD expert model

APAD MicroSaint Expert Model: Graphic Representations

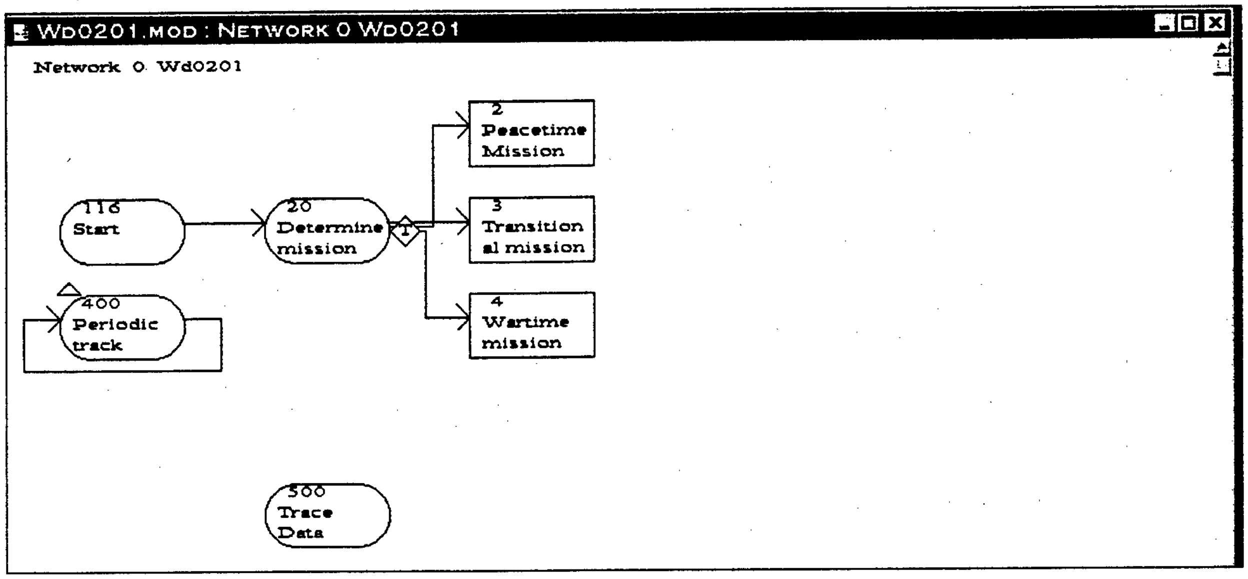


Figure A-1. Top Level APAD Expert Model

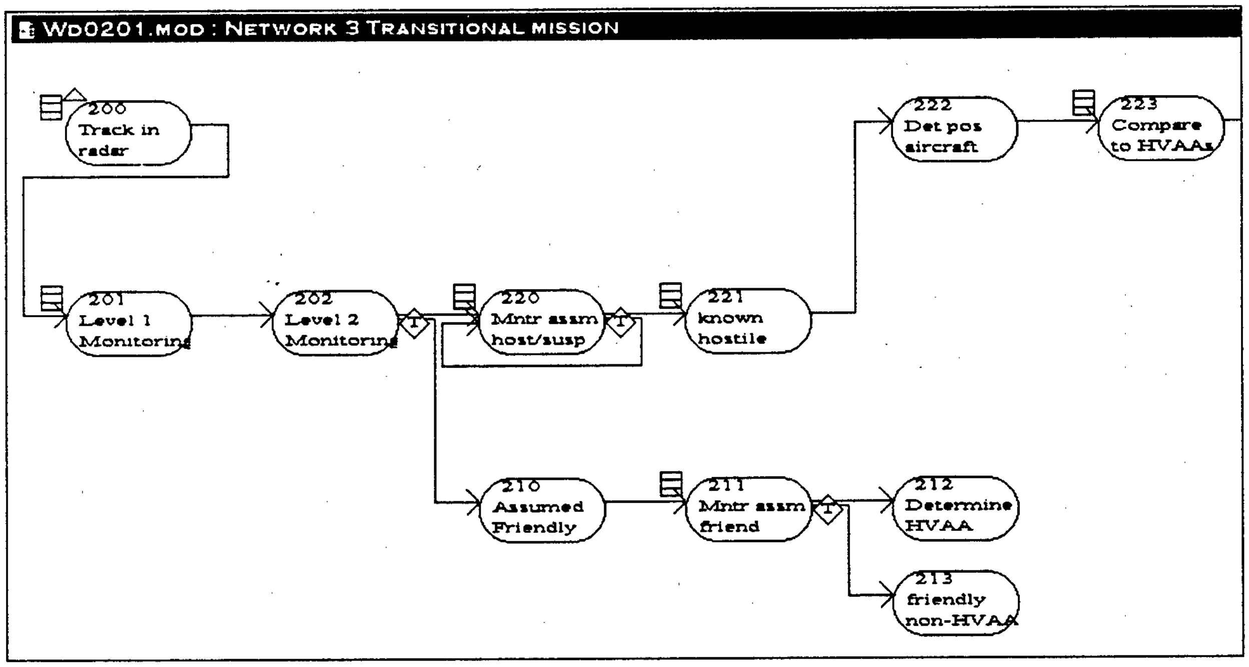


Figure A-2. APAD Expert Model, Part 1

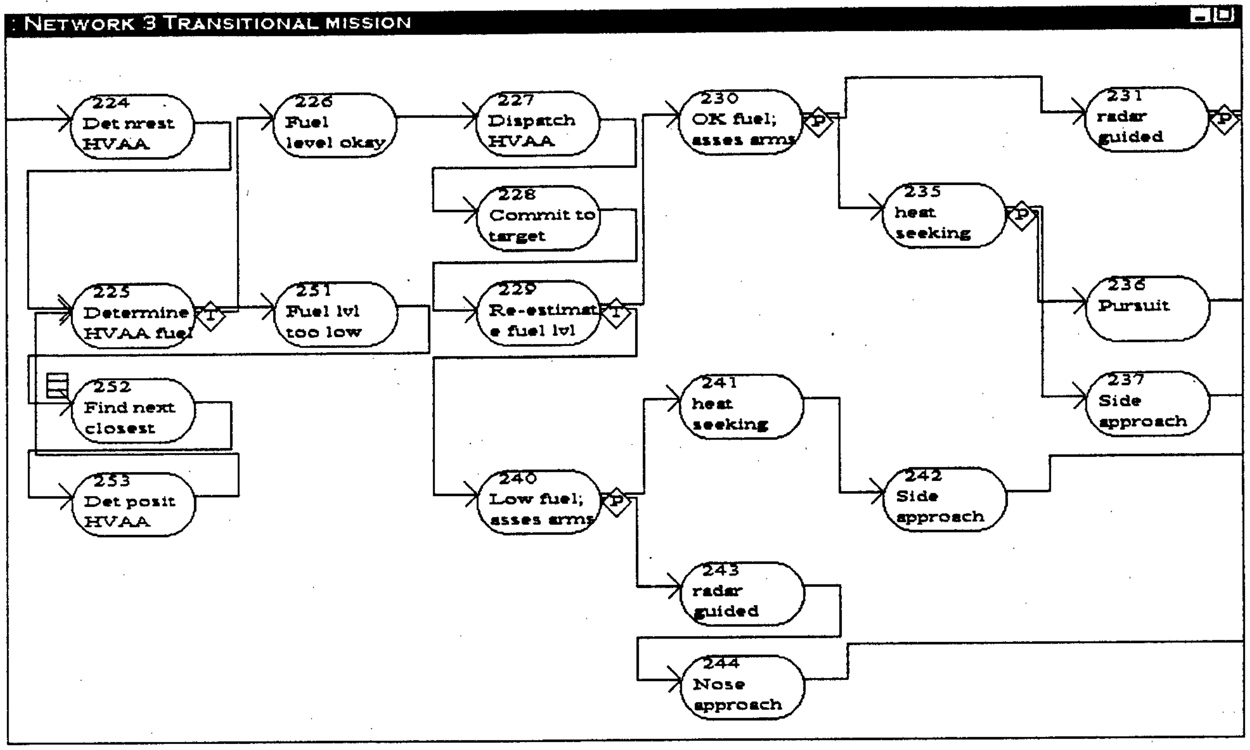


Figure A-3. APAD Expert Model, Part 2

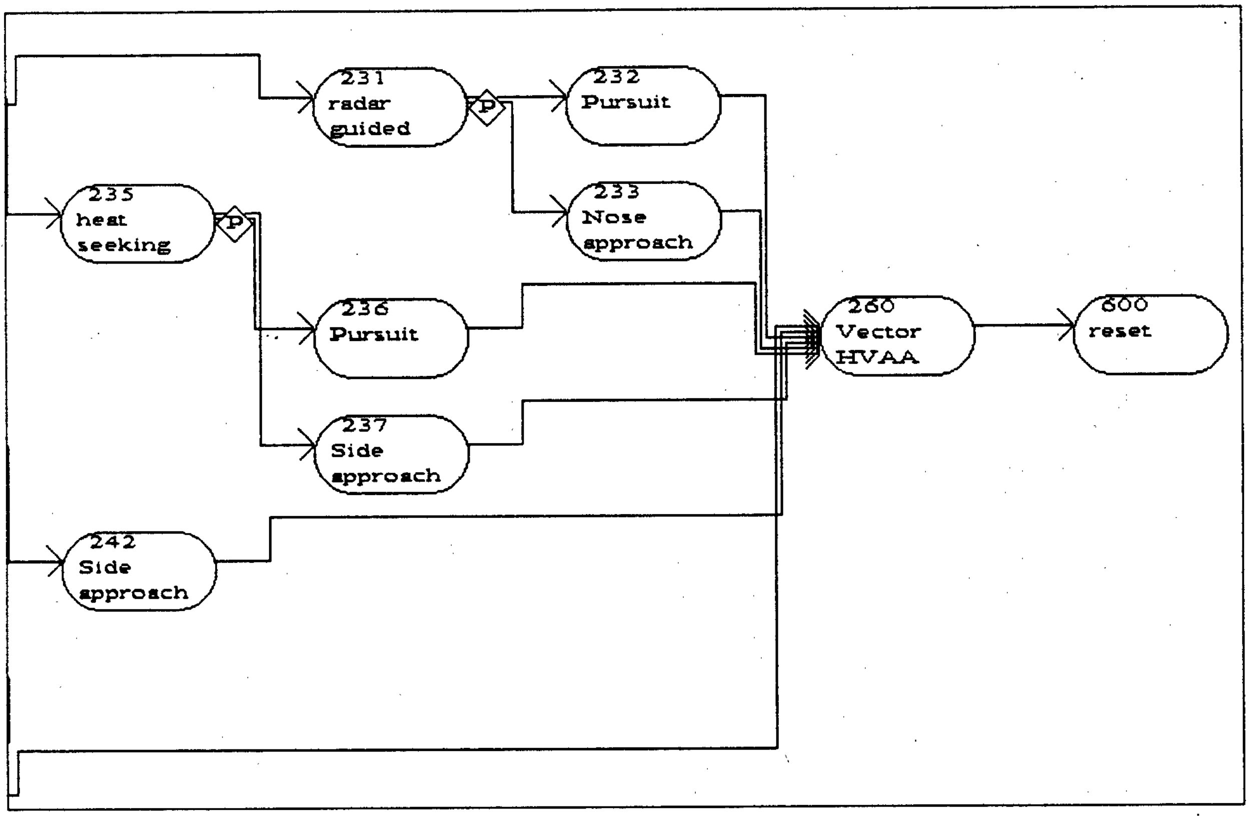


Figure A-4. APAD Expert Model, Part 3

APAD Expert Model MicroSaint Code

19980603 Release 3.0 Build A with ActionView and OptQuest

```
400
+++++
2
2
Peacetime Mission
0
1;
---
---
0
2
0
1
1
1


113
1
+++++
1
5
Determine HVAA fuel level
2
1;
---
Comptency:  Ability to determine fuel level based on HVAA time in air and rate of fuel consumption
---
2
26
71
5
3
1
1
1


1
{DECIDE;}
---
---
4
1,{air_time[tag]=2;}
---
1,{air_time[tag]=1;}
---
---
---
{if Air_time[tag]<360, then fuel_leve[tag]]:=1, else fuel_level[tag]:=2;}
---
1
+++++
1
9
Monitor
2
1;
---
---
1
74
2
```

2
1
1
1

1

2
1;

1
+++++
1
12
Detemine friendly non-HVAA
2
1;

Comptency: Categorization, Symbology knowledge

2
74
9
0
4
1
1
1

1

4
1;

1;

{Num_Friend:=Num_Friend+1;}

1
+++++
1
17
Send HVAA for refuel
2
1;

Competency: Awarenss of refuel requirements

1
73
7
2
1
1
1

1

2

```
1;
---
---
---
{Num_HVAA:=Num_HVAA-1;}
---
1
+++++
1
19
Determine HVAA
2
1;
---
Competency:  Categorization, Symbology knowledge
---
1
5
2
1
1
1
1


1
---
---
2
1;
---
---
---
{Num_HVAA:=Num_HVAA+1;}
---
1
+++++
1
21
Hostility type?
2
1; {ops_avail[WD]>=1 & tracktot <=0;}
---
Competencies:  Symbology recognition, categorization, and global awareness
---
2
19
12
0
2
1
1
1


1
---
---
4
1;{host_type==2;}
---
1;{host_type==1;}
---
{ops_avail[WD]-=1;
processing[WD]:=tag;}
---
---
{tracknum:=Track_Num[tag%1000];
current:=tag;
WDprior:=WD_priorit Y[tag%1000];}
```

1
+++++
1
26
Fuel low
2
1;

Competency: Correct determination of fuel level

1
17
6
2
1
1
1

1

2
1;

1
+++++
1
71
Fuel okay
2
1;

Competency: Correct judgement of fuel level

1
77
6
3
1
1
1

1

2
1

1
+++++
1
73
Determine next closest HVAA
2
1;

Competency: Distance calculation

1
5
8

2
1
1
1

1

2
1;

1
+++++
1
74
Need to escort?
2
1;

Competency: Mission awareness, planning

2
75
76
1
5
1
1
1

1

4
1;

1;

1
+++++
1
75
Determine closest HVAA
2
1;

Competency Distance calculation from data or screen view, Computation skills, visual comparison
---
1
5
3
4
1
1
1

1

2
1;

```
---
---
---
---
1
+++++
1
76
Monitor
2
1;
---
Competency:  Situational awareness
---
1
74
1
7
1
1
1


1
---
---
2
1;
---
---
---
---
1
+++++
1
77
Send escort
2
1;
---
measureable--commit action of both aircraft

competency   final commit decision
---
0
7
3
1
1
1


1
{2*MOVE_CUR_T}
---
---
1
---
---
{Num_HVAA=Num_HVAA-1;}
---
1
+++++
1
113
Track in system
2
1,
---
No competency--system issue
```

```
---
1
21
0
1
1
1
1


1
---
---
2
1;
---
---
---
---
1
+++++
2
3
Transitional mission
0
1;
---
---
0
2
1
1
1
1


200
1
+++++
1
200
Track in radar
3
ac_id[tag] < 9999;
---
---
1
201
0
1
1
1
1


1
---
---
2
1;
---
---
---
---
1
+++++
1
201
Level 1 Monitoring
3
```

```
WD > 0 & ac_id[tag]>121;
{unknown assume friend or host/susp}
---
Competency:  Monitoring
---
1
202
0
3
1
1
1


1
.1;  {6 seconds}
---
---
2
ac_id[tag]>121;
{unknown assume friend or host/susp}
---
---
---
comp_1:=1;
comp_2:=0;
comp_3:=0;

---
1
+++++
1
202
Level 2 Monitoring
3
1,
---
Competency:  Monitoring
---
2
220
210
1
3
1
1
1


1
---
---
4
(ac_id[tag]>=141 & ac_id[tag]<=148);
{id # from 141 to 148 assume host/susp}
---
(ac_id[tag]>=130 & ac_id[tag]<=140) | (ac_id[tag]==211 | ac_id[tag]==213 | ac_id[tag]==415);
{id # 130 to 140 assume friend}
---
---
---
comp_1:=1,
comp_2:=0,
comp_3:=0;

---
1
+++++
1
210
```

```
Assumed Friendly
3
1;
---
---
1
211
2
5
1
1
1


1
---
---
2
1;
---
---
---
tot_friendly+=1;
Friendly[tot_friendly] := tag;
---
1
+++++
1
211
Mntr assm friend
3
WD > 0 & ((ac_id[tag]==211 | ac_id[tag]==415) | ac_id[tag]==213);
{id # 130 to 140 assume friend}
---
Competency: Monitoring
---
2
212
213
3
5
1
1
1


1
.1, {6 seconds}
---
---
4
ac_id[tag]==211 | ac_id[tag]==415; {friendly general notspecified - we are using for HVAA or 415 for AWACS}
---
ac_id[tag]==213; {friendly general nonmil Civilian - we are using for COMAIR}
---
---
---
comp_1:=1,
comp_2:=0,
comp_3:=0,

---
1
+++++
1
212
Determine HVAA
3
1;
---
```

```
Competency:  Categorization, Symbology knowledge
---
0
4
5
1
1
1


1
CLASS_MATCH;
---
---
1
---
---
if Miss_type1[tag]==1 |
  Miss_type2[tag]==1
then tot_HVAA+=1,
      avail_HVAA+=1,
      HVAA[tot_HVAA] := tag;

comp_1:=2;
comp_2:=4;
comp_3:=0;

---
1
+++++
1
213
friendly non-HVAA
3
1,
---
Competency:  Categorization, Symbology knowledge
---
0
4
6
1
1
1


1
CLASS_MATCH;
---
---
1
---
---
comp_1:=2;
comp_2:=4;
comp_3:=0;

---
1
+++++
1
220
Mntr assm host susp
3
WD > 0,
---
Competency:  Monitoring, ROE use
---
2
220
```

```
221
2
3
1
1
1


1
.2;  {12 seconds}
---
---
4
truncate(ac_id[tag]/10)==14;
{id # from 141 to 148 assume host/susp}
---
ac_id[tag]==311 & hostile_act[tag] ==1; {hostile}
---
---
---
HOSTILE_ACT;
if hostile_act[tag] ==1
then ac_id[tag]:=311;

comp_1:=1;
comp_2:=3;
comp_3:=0;

---
1
+++++
1
221
known hostile
3
WD>0;
---
Competency:  Categorization, Symbology knowledge
---
1
222
3
3
1
1
1


1
CLASS_MATCH;
---
---
2
1,
---
WD-=1;
---
---
host_count+=1;
com_hostile := Squad_Id[tag];
comp_1:=2;
comp_2:=4;
comp_3:=0;
TRACE;
---
1
+++++
1
222
Det pos aircraft
```

```
3
1;
---
Competency:  Data gathering, system use
---
1
223
4
1
1
1
1


1
.5/60; {half a second}
---
---
2
1;
---
---
---
comp_1:=5;
comp_2:=6;
comp_3:=0;
TRACE;
---
1
+++++
1
223
Compare to HVAAs
3
avail_HVAA>0;
---
Competency:  Distance calculation from data or screen view, Computation skills, visual comparison
---
1
224
5
1
1
1
1


1
.5/60; {half a second}
---
---
2
1;
---
---
---
comp_1:=7;
comp_2:=8;
comp_3:=9;
TRACE;
---
1
+++++
1
224
Det nrest HVAA
3
1;
---
Competency:  Situational awareness, Planning
```

```
---
1
225
6
1
1
1
1


1
.5/60; {half a second}
---
---
2
1;
---
DET_NEAREST;
if do_compare then COMP_HVAA;
---
---
comp_1:=10;
comp_2:=11;
comp_3:=0;
TRACE;
---
1
+++++
1
225
Determine HVAA fuel level
3
1;
---
Competency:  Knowledge of heuristic, Situational awareness
---
2
251
226
8
1
1
1
1


1
.5/60; {half a second}
---
---
4
fuel_ok==0;
---
fuel_ok==1;
---
---
---
FUEL_CHECK1;
if fuel_ok==0 then occup[nearest]:=2, avail_HVAA-=1;

comp_1:=12;
comp_2:=10;
comp_3:=0;
TRACE;

---
1
+++++
1
226
```

```
Fuel level okay
3
1;
---
Competency:  Knowledge of heuristic, Situational awareness
---
1
227
9
1
1
1
1


1
---
---
2
1;
---
---
---
comp_1:=12;
comp_2:=10;
comp_3:=0;
TRACE;

---
1
+++++
1
227
Dispatch HVAA
3
1;
---
Competency:  Aircraft selection

Measure:  Commit action
---
1
228
10
1
1
1
1


1
SPEAK;
---
---
2
1;
---
---
---
com_hvaa1 :=Squad_Id[nearest];
com_hvaa2 :=Squad_Id[nearest];
DISPATCH;


comp_1:=13;
comp_2:=0;
comp_3:=0;
TRACE;

---
```

```
1
+++++
1
228
Commit to target
3
1;
---
Competency:  Decision-making

Observable:  Commit action on track
---
1
229
11
1
1
1
1


1
PUSH_BUTTON;
---
---
2
1;
---
---
---
com_time := clock;
commit_count+=2;

comp_1:=14;
comp_2:=0;
comp_3:=0;
TRACE;

---
1
+++++
1
229
Re-estimate fuel lvl
3
1;
---
---
2
240
230
12
1
1
1
1


1
.5 60. {half a second}
---
---
4
fuel_ok==0;
---
fuel_ok==1;
---
---
---
FUEL_CHECK2;
```

```
---
1
+++++
1
230
OK fuel; asses arms
3
1;
---
Competency:  Knowledge of heuristic, Situational awareness
---
2
235
231
15
1
1
1
1


1
.5/60; {half a second}
---
---
3
Miss_type2[nearest]>0;
---
Miss_type1[nearest]>0;
---
---
---
comp_1 =12.
comp_2 =10.
comp_3 =0.
TRACE.

---
1
+++++
1
231
radar guided
3
1.
---
Competency  Knowledge of heuristic, Situational awareness, knowledge of assets
---
2
233
232
16
1
1
1
1


1
DECIDE.
---
---
3
1.
---
1.
---
---
---
```

```
comp_1:=12;
comp_2:=10;
comp_3:=15;
TRACE;

---
1
+++++
1
232
Pursuit
3
1;
---
Competency:  Knowledge of heuristic, Situational awareness
---
1
260
17
1
1
1
1


1
DECIDE;
---
---
2
1;
---
---
---
vector:=1;
comp_1:=12;
comp_2:=10;
comp_3:=0;
TRACE;

---
1
+++++
1
233
Nose approach
3
1.
---
Competency.  Knowledge of heuristic, Situational awareness
---
1
260
17
2
1
1
1


1
DECIDE;
---
---
2
1;
---
---
---
vector:=2;
```

```
comp_1:=12;
comp_2:=10;
comp_3:=0;
TRACE;

---
1
+++++
1
235
heat seeking
3
1;
---
Competency:  Knowledge of heuristic, Situational awareness, knowledge of assets
---
2
237
236
16
3
1
1
1


1
DECIDE;
---
---
3
1;
---
1;
---
---
---
comp_1:=12;
comp_2:=10;
comp_3:=15;
TRACE;

---
1
+++++
1
236
Pursuit
3
1;
---
Competency:  Knowledge of heuristic, Situational awareness
---
1
260
17
3
1
1
1


1
DECIDE;
---
---
2
1;
---
---
```

```
---
vector:=1;
comp_1:=12;
comp_2:=10;
comp_3:=0;
TRACE;

---
1
+++++
1
237
Side approach
3
1;
---
Competency:  Knowledge of heuristic, Situational awareness
---
1
260
17
4
1
1
1


1
DECIDE;
---
---
2
1;
---
---
---
vector:=3;
comp_1:=12;
comp_2:=10;
comp_3:=0;
TRACE;

---
1
+++++
1
240
Low fuel, asses arms
3
1;
---
Competency:  Knowledge of heuristic, Situational awareness
---
2
241
243
15
5
1
1
1


1
.5 60; {half a second}
---
---
3
Miss_type2[nearest]>0;
---
```

```
Miss_type1[nearest]>0;
---
---
---
comp_1:=12;
comp_2:=10;
comp_3:=0;
TRACE;

---
1
+++++
1
241
heat seeking arms
3
1;
---
Competency:  Knowledge of heuristic, Situational awareness, knowledge of assets
---
1
242
16
6
1
1
1


1
DECIDE;
---
---
2
1;
---
---
---
comp_1:=12;
comp_2:=10;
comp_3:=15;
TRACE;

---
1
+++++
1
242
Side approach
3
1;
---
Competency:  Knowledge of heuristic, Situational awareness
---
1
260
17
6
1
1
1


1
DECIDE;
---
---
2
1;
---
```

```
---
---
vector:=3;
comp_1:=12;
comp_2:=10;
comp_3:=0;
TRACE;

---
1
+++++
1
243
radar guided arms
3
1;
---
Competency:  Knowledge of heuristic, Situational awareness, knowledge of assets
---
1
244
16
7
1
1
1


1
DECIDE;
---
---
2
1;
---
---
---
comp_1:=12;
comp_2:=10;
comp_3:=15;
TRACE;

---
1
+++++
1
244
Nose approach
3
1;
---
Competency  Knowledge of heuristic, Situational awareness
---
1
260
17
7
1
1
1


1
DECIDE;
---
---
2
1;
---
---
```

```
---
vector:=2;
comp_1:=12;
comp_2:=10;
comp_3:=0;
TRACE;

---
1
+++++
1
251
Fuel lvl too low
3
1;
---
Competency:  Knowledge of heuristic, Situational awareness
---
1
252
9
3
1
1
1


1
---
---
2
1;
---
---
---
comp_1:=12;
comp_2:=10;
comp_3:=0;
TRACE;

---
1
+++++
1
252
Find next closest HVAA
3
avail_HVAA>0;
---
Competency:  Situational awareness, decision-making
---
1
253
8
4
1
1
1


1
.5/60; {half a second}
---
---
2
1;
---
DET_NEAREST;
---
---
```

```
comp_1:=10;
comp_2:=14;
comp_3:=0;
TRACE;

---
1
+++++
1
253
Det posit HVAA
3
1;
---
Competency:  Data gathering, system use
---
1
225
9
4
1
1
1


1
.5/60; {half a second}
---
---
2
1;
---
---
---
comp_1:=5;
comp_2:=6;
comp_3:=0;
TRACE;

---
1
+++++
1
260
Vector HVAA
3
1;
---
Competency:  Knowledge of rules for HVAA use given fuel and armament status

Observeable that indicates success at next two levels up:  Measured by vector taken by committed HVAA to target
---
1
600
18
3
1
1
1


1
SPEAK;
---
---
2
1;
---
if do_compare then index:=commit_count-1, CIE_DATA, start(601, 0);
---
```

```
---
WD+=1;
comp_1:=16;
comp_2:=0;
comp_3:=0;
TRACE;
if do_compare then index:=commit_count, CIE_DATA, start(601,0);
---
1
+++++
1
600
reset
3
1;
---
---
0
19
3
1
1
1


1
---
---
1
---
---
com_hvaa1:=0;
com_hvaa2:=0;
com_hostile:=0;
not_nearest:=0;
not_fuel1:=0;
---
1
+++++
1
601
CIE Data Gathering
3
1;
---
---
0
19
4
1
1
1


1
---
---
1
---
---
---
1
+++++
2
4
Wartime mission
0
1;
---
---
```

0
2
2
1
1
1

115
1
+++++
1
53
Determine position
4
1;

Competency: Data gathering, system use

1
56
3
2
1
1
1

1

2
1;

1
+++++
1
54
Determine known hostile
4
5,

Competency: Categorization, Symbology knowledge

1
57
2
4
1
1
1

1

2
1.

1
+++++
1
55
Determine potent hostile air

```
4
1;
---
Competency:  Categorization, Symbology knowledge
---
1
57
2
5
1
1
1


1
---
---
2
1;
---
---
---
---
1
+++++
1
56
Determine HVAA armament
4
1;
---
Competency:  Knowledge of assets
---
2
101
102
4
2
1
1
1


1
---
---
4
1;{armament_type==1;}
---
1;{armament_type==2;}
---
---
---
---
1
+++++
1
57
Determine position
4
1;
---
Competency:  Data gathering, system use
---
1
58
3
4
1
1
```

1

1

2
1;

1
+++++
1
58
Compare posit to all HVAAs
4
1;

Competency: Distance calculation from data or screen view, Computation skills, visual comparison
---
1
59
4
4
1
1
1

1

2
1;

1
+++++
1
59
Determine nearest HVAA
4
1;

Competency: Situational awareness, Planning

1
63
5
4
1
1
1

1

2
1;

1
+++++

1
60
Monitor
4
1;

Competency: Situational awareness

1
82
4
6
1
1
1

1

2
1;

1
+++++
1
61
Detemine friendly non-HVAA
4
1,

Competency: Categorization, Symbology knowledge

1
82
2
6
1
1
1

1

2
1,

1
+++++
1
63
Check HVAA fuel level
4
1.

Competency Knowledge of heuristic, Situational awareness

2
111
110
6
4
1

1
1

1

4
1;

1;

1
+++++
1
64
Dispatch HVAA
4
1;

Competency: Decision-making, system use

Observable: HVAA commit action and target commit action; verbal comm with pilot
---
0
9
2
1
1
1

1

1

1
+++++
1
82
Assess for support
4
1;

Competency: Situational awareness

2
60
57
3
6
1
1
1

1

4
1;

1;

1
+++++
1
90
Hostility level?
4
1;

Competency: Symbol recognition, categorization, and global awareness
---
4
54
55
61
100
1
3
1
1
1

1

4
1;

1;

1;

1;

---
tag:=tag+1;

1
+++++
1
100
Determine HVAA
4
1;

Competency: Categorization, Symbology knowledge

1
53
2
2
1
1
1

1

2
1;

1

+++++
1
101
Radar guided armament
4
1;

Competency: Knowledge of assets

1
104
5
1
1
1
1

1

2
1;

1
+++++
1
102
Heat-seeking armament
4
1,

Competency: Knowledge of assets

1
103
5
2
1
1
1

1

2
1,

1
+++++
1
103
Send home
4
1,

Competency: Situational awareness
Observable: Call button engage, pilot/flight changes course to base
---
0
6
2
1

1
1

1

1

1
+++++
1
104
Check fuel level
4
1;

Competency: Knowledge of heuristic

2
106
105
6
1
1
1
1

1

4
1;

1;

1
+++++
1
105
Fuel okay
4
1;

Competency: Correct use of heuristic

observable length of time since last refueling

1
107
7
0
1
1
1

1

2
1;

1
+++++
1
106
Fuel low
4
1;

Competency: Correct use of heuristic

observable: length of time since last refueling

1
108
7
1
1
1
1

1

2
1;

1
+++++
1
107
Okay to commit
4
1;

Competency: Decision-making

Observable: AC stays in pattern

0
8
0
1
1
1

1

1

1
+++++
1
108
Send for re-fueling
4
1;

Competency: Decision-making

observable: comm between WD and pilot, change in AC heading, toward tanker, call-in to tanker WD

0
8
1
1
1
1

1

1

1
+++++
1
109
Commit to target
4
1;

Competency: Decision-making

1
64
8
2
1
1
1

1

2
1,

1
+++++
1
110
Fuel too low
4
1,

Competency Correct use of heuristic

observable length of time since last refueling

1
112
7
4
1
1
1

1

2
1;

```
---
---
---
---
1
+++++
1
111
Fuel okay
4
1;
---
Competency:  Correct use of heuristic

observable:  length of time since last refueling
---
1
109
7
2
1
1
1


1
---
---
2
1;
---
---
---
---
1
+++++
1
112
Determine next nearest HVAA
4
1;
---
Competency:  Decision-making, situational awareness
---
1
63
8
4
1
1
1


1
---
---
2
1;
---
---
---
---
1
+++++
1
115
Track in system
4
1;
---
```

```
---
1
90
0
3
1
1
1


1
---
---
2
1;
---
---
---
---
1
+++++
1
20
Determine mission type
0
1;
---
Competency:  Understanding the overall mission
---
3
2
3
4
1
1
1
1
1


1
---
---
4
mission_type==1;
---
mission_type==2;
---
mission_type==3;
---
---
---
---
1
+++++
1
116
Start
0
1;
---
---
1
20
0
1
1
1
1
```

```
1
---
---
2
1;
---
---
---
---
1
+++++
1
400
Periodic track update
0
1;
---
---
1
400
0
2
1.
1
1


1
0.2; {12 seconds - same as radar sweep of an AWACS}
---
---
2
1,
---
---
---
count:=1;
while count <= tot_aircraft do
current := aircraft[count],
UPDATE_POS.
count+=1;


---
1
+++++
1
500
Trace Data Gathering
0
1.
---
---
0
1
4
1
1
1


1
---
---
1
---
---
---
1
```

```
+++++
********
1
7

223
1
---
---
---
+++++
1
21

21
1
---
---
---
+++++
1
71

90
1
---
---
---
+++++
1
87

252
1
---
---
---
+++++
1
114

200
3
ac_id[tag] < 9999;
---
---
---
+++++
1
123

201
3
ac_id[tag]>121;
---
---
---
+++++
1
124

211
3
((ac_id[tag]==211 | ac_id[tag]==415) | ac_id[tag]==213);
---
---
---
+++++
1
125
```

```
220
1
---
---
---
+++++
1
221

221
1
---
---
---
+++++
********
ac_id
AC identification number for each AC by tag
3
1
500
0.000000
+++++
ackn
Acknowledgement of request given
1
0.000000
+++++
adding
Restricts addition of new tracks
1
0.000000
+++++
air_time
keep total time HVAA has been in air
4
1
300
0.000000
+++++
aircraft
keeps track of aircraft tags
3
1
50
0.000000
+++++
Altitude
AC altitude in feet by tag
4
1
500
0.000000
+++++
altitudein
altitude of a new track
1
0.000000
+++++
amplin

1
0.000000
+++++
amplop

1
0.000000
+++++
arr_mod_hook

2
0.000000
+++++
arr_mod_id
Latency for identificatio by model
2
0.000000
+++++
arr_op_hook

2
0.000000
+++++
arr_op_id
Latency between arrival an optr ident
2
0.000000
+++++
arrival
Time a new track arrived
2
0.000000
+++++
atime
Time of request acknowledgement
2
0.000000
+++++
attacking
track attacking hostiles
3
1
300
0.000000
+++++
avail_HVAA
HVAA resource counter
1
0.000000
+++++
bearin

2
0.000000
+++++
Bearing
bearing of track
4
1
250
0.000000
+++++
bearing1
bearing from one AC to another
2
0.000000
+++++
bearing2
bearing from one AC to another
2
0.000000
+++++
bearop

1
0.000000
+++++
bracket1
```

```
one edge of a range of bearings
2
0.000000
+++++
bracket2
one edge of a range of bearings
2
0.000000
+++++
cie_arm
CIE value for armament of HVAA chosen by o
1
0.000000
+++++
cie_fuel2
CIE value for fuel check 2 type
1
0.000000
+++++
cie_hvaa
CIE value for hvaa type for CIE output file
1
0.000000
+++++
cie_vector
CIE value for vector chosen by op
1
0.000000
+++++
clock
System Variable
2
0.000000
+++++
cn01

1
0.000000
+++++
cnnom

1
0.000000
+++++
com_count
Communications counter
1
0.000000
+++++
com_host_op
Squad # of hostile committed by operator
3
1
50
0.000000
+++++
com_hostile
Squad num of hostile that was committed
1
0.000000
+++++
Com_Hvaa_Op
Squad # of committed hvaa by operator
3
1
50
0.000000
+++++
com_hvaa1
Squad number of the committed hvaa
```

```
1
0.000000
+++++
com_hvaa2
Squad number of the committed hvaa
1
0.000000
+++++
com_time
the time of the committ
2
0.000000
+++++
com_time_op
the time of committ by the operator
4
1
50
0.000000
+++++
commit_count
count the commits
1
0.000000
+++++
comp_1
the first competency listed for a task
1
0.000000
+++++
comp_2
the second competency listed for a task
1
0.000000
+++++
comp_3
the third competency listed for a task
1
0.000000
+++++
corr
Correlation
1
0.000000
+++++
count
generic counter
1
0.000000
+++++
Course

4
1
500
0.000000
+++++
course_known

3
1
250
0.000000
+++++
coursein

2
0.000000
+++++
courseop
```

```
2
0.000000
+++++
CPA_range
range of hostile ship CPA
4
1
300
0.000000
+++++
ctype
Communication type
1
0.000000
+++++
current
Curent tag of interest
1
0.000000
+++++
dbugon

1
0.000000
+++++
delta_alt
altitude delta between two aircraft
2
0.000000
+++++
dnetin

1
0.000000
+++++
do_compare
flag for comparing exp vs. oper actions
1
0.000000
+++++
drop

2
0.000000
+++++
dsnsin

1
0.000000
+++++
dsnsop

1
0.000000
+++++
duration
System Variable
2
0.000000
+++++
emit
Emitter reqort sent
1
0.000000
+++++
Emit_req
request for emitter on a track
3
1
```

```
300
0.000000
+++++
Emit_stat
Is emitter correlated with track
3
1
250
0.000000
+++++
Emit_typ
Type of emitter associated with track
3
1
300
0.000000
+++++
emitreq
Emitter request sent
1
0.000000
+++++
emitstat

1
0.000000
+++++
engage
variable to track whether or not to engage
3
1
300
0.000000
+++++
escort

3
1
300
0.000000
+++++
ESM

3
1
250
0.000000
+++++
ESM_known

3
2
300
5
0.000000
+++++
esm_set
to keep track of model setting esm info
3
1
300
0.000000
+++++
estatin
friendly(1), commercial (2)or hostile (3)esm
1
0.000000
+++++
etyp
```

```
1
0.000000
+++++
etypin

2
0.000000
+++++
finalp
final priority
1
0.000000
+++++
found_track
Track found in search
1
0.000000
+++++
Friendly
keeps which tags are friendly AC
3
1
100
0.000000
+++++
from
array index for x,y position
1
0.000000
+++++
fuel_level
estimated fuel level of HVAA
3
1
500
0.000000
+++++
fuel_ok
flag for fuel assessment routing condition
1
0.000000
+++++
hook_count

3
1
10
0.000000
+++++
hook_time
Time a hook was made
2
0.000000
+++++
host_count
counts how many hostiles are attended to
1
0.000000
+++++
Host_tracks
Number of HVAA tracks
1
0.000000
+++++
host_type
set true hostility of AC in scenario
1
0.000000
+++++
Host_type
```

```
set true hostility of AC in scenario
3
1
100
0.000000
+++++
hostile_act
flag for hostile act by an AC
3
1
500
0.000000
+++++
HVAA
keeps which tags are HVAA
3
1
50
0.000000
+++++
hvaa_op
the hvaa chosen by the operator
1
0.000000
+++++
HVAA_tracks
Number of HVAA tracks
1
0.000000
+++++
i

1
0.000000
+++++
i_cat
# of dimens on which tracks are compared
1
0.000000
+++++
i_cat2
# dimens to check for info
1
0.000000
+++++
ID
ID number for AC in scenario events
1
0.000000
+++++
id_count
Counter used in identification
1
0.000000
+++++
Identified
Time track was identified
4
1
250
0.000000
+++++
IDff
idetnify friend or foe
3
1
250
0.000000
+++++
idffin
```

```
2
0.000000
+++++
idffop

1
0.000000
+++++
IDplat

2
0.000000
+++++
IDplatf
Type of platform
3
1
250
0.000000
+++++
idplatin
type of platform - new track in
1
0.000000
+++++
idplatop

1
0.000000
+++++
idtypin

2
0.000000
+++++
idtypop

1
0.000000
+++++
IFF_known

3
2
300
5
0.000000
+++++
Iff_mode
Mode of trandponder
3
1
250
0.000000
+++++
iff_set

3
1
300
0.000000
+++++
iffmodin

2
0.000000
+++++
iffmodop
```

```
1
0.000000
+++++
iffp

1
0.000000
+++++
index
index to operator data arrays
1
0.000000
+++++
keyin
Input buffer for keypress
1
0.000000
+++++
Last_Upd
Last time position was updated
4
1
500
0.000000
+++++
maxa

1
0.000000
+++++
Maxspd
Array of max speeds
3
1
250
0.000000
+++++
min_range
Use to keep the min range to a target
2
0.000000
+++++
mina

1
0.000000
+++++
minin
minutes in to convert time
1
0.000000
+++++
miss_ty_1
flag for missile type radar
1
0.000000
+++++
miss_ty_2
flag for missile type heat seaker
1
0.000000
+++++
Miss_type1
flag for radar missile for each AC by tag
3
1
500
0.000000
+++++
Miss_type2
```

```
flag for heat seak missile for each AC by tag
3
1
500
0.000000
+++++
mission_type
set type of mission
1
0.000000
+++++
mod_hook

2
0.000000
+++++
mspd
max speed in
1
0.000000
+++++
n
number of positions in pattern flight
1
0.000000
+++++
nearest
tag value for HVAA nearest target
1
0.000000
+++++
next_point
next point counter in pattern flight
3
1
500
0.000000
+++++
no_emitter
No emitter report given
2
0.000000
+++++
not_done
generic flag for logic statements
1
0.000000
+++++
not_fuel1
flag for not enough fuel in hvaa chosen by op
1
0.000000
+++++
not_hostile
boolean for hostile action flag
1
0.000000
+++++
not_nearest
flag for op HVAA that is not nearest
1
0.000000
+++++
Num_AssuHos
track number of assumed hostile or unk assu
1
0.000000
+++++
Num_Friend
track number of friendlies
```

```
1
0.000000
+++++
Num_Hostile
track number of hostile
1
0.000000
+++++
Num_HVAA
keep count of number of HVAAs
1
0.000000
+++++
num_tracks
number of tacks in the scenrio
1
0.000000
+++++
numfixes
number of eye fixations during search
1
0.000000
+++++
objective
System Variable
2
0.000000
+++++
occup
status of any AC
3
1
500
0.000000
+++++
OCFrom
Operator communcation from
4
1
500
0.000000
+++++
OCReq
Communcation requested by another oprtr
3
1
1000
0.000000
+++++
OCTime
Operator communcations time
4
1
500
0.000000
+++++
OCTo
Operator communcation TO
3
1
500
0.000000
+++++
OCTrack
Track OPERATOR refers to in communication
3
1
1000
0.000000
+++++
```

OCType
Operator communcation type
3
1
500
0.000000
+++++
OKey
Keypress entered by an operator
3
2
5
1000
0.000000
+++++
On_deck
is ship based A/C on deck?
3
1
200
0.000000
+++++
ondkin
is a/c on deck?
1
0.000000
+++++
op_hook

2
0.000000
+++++
op_id
Time operator id'd a track
2
0.000000
+++++
op_mod_hook

2
0.000000
+++++
op_mod_id
Time relative to model for operator id
2
0.000000
+++++
ops_avail
Number of operators of a type available
3
1
10
0.000000
+++++
optr
Operator assigned to a task
2
0.000000
+++++
optrin
Input buffer for operator number
1
0.000000
+++++
optrout
Operator sending a message
1
0.000000
+++++
optrrec

```
Operator recieveing a message
1
0.000000
+++++
OReq
Operator request for the track
3
1
10000
0.000000
+++++
Origin
Identify country of origin
3
1
250
0.000000
+++++
originin

2
0.000000
+++++
OTime
Time an operator action occurred
4
2
5
1000
0.000000
+++++
OTrack
Track number operator action is on
3
2
5
1000
0.000000
+++++
otstep
Time step for actual operator data
3
1
5
0.000000
+++++
pattern
boolean flag if AC is flying a pattern
1
0.000000
+++++
Pattern
pattern flag for each AC by tag
3
1
500
0.000000
+++++
Positions
number of positions for pattern flight by tag
3
1
500
0.000000
+++++
prev_point
previous point counter in pattern flight
3
1
500
```

```
0.000000
+++++
prioritytest
snapshot  test variable-15
2
0.000000
+++++
processing
Tag currently being processed
3
1
250
0.000000
+++++
quadin

1
0.000000
+++++
quadop

1
0.000000
+++++
Quadrant
quadrant of arrival for track
3
1
250
0.000000
+++++
Range
Range from the ship
4
1
250
0.000000
+++++
range
range between two aircraft in three dimensio
2
0.000000
+++++
rangein

2
0.000000
+++++
rangeoop

1
0.000000
+++++
requested
Communication was/was not requested
1
0.000000
+++++
ROE

2
0.000000
+++++
run
System Variable
1
0.000000
+++++
secsin
seconds in to convert time
```

```
1
0.000000
+++++
seed
System Variable
1
1.000000
+++++
sin

1
0.000000
+++++
Speed
Speed of track
4
1
500
0.000000
+++++
speedin

2
0.000000
+++++
speedop

2
0.000000
+++++
Squad_count
Number of AC in squad by squad id num
3
1
10000
0.000000
+++++
Squad_Id
Squad ids for each aircraft by tag
3
1
500
0.000000
+++++
squad_num
number of AC in squad
1
0.000000
+++++
squadin
squad id number
1
0.000000
+++++
stat

1
0.000000
+++++
statin

2
0.000000
+++++
statop

2
0.000000
+++++
statpop
```

```
1
0.000000
+++++
t_to_point
time to fly to next point
2
0.000000
+++++
tag
System Variable
1
0.000000
+++++
tagin
Input buffer for tag numbers
1
0.000000
+++++
task_num
the number of a task
1
0.000000
+++++
temp
temporary storage variable
2
0.000000
+++++
temp1
temporary integer variable
1
0.000000
+++++
temp2
temporary integer variable
1
0.000000
+++++
testIDff

2
0.000000
+++++
testIDtype

2
0.000000
+++++
testlvl

2
0.000000
+++++
TIC
Constant to identify TIC
1
0.000000
+++++
TIC_toe

2
0.000000
+++++
Time_In
Time of track arrival
4
1
500
0.000000
```

```
+++++
time_step
time for updating of current flight interval
2
0.000000
+++++
timein
Input buffer for times
2
0.000000
+++++
to
array index for xy position
1
0.000000
+++++
tot_aircraft
counts the total aircraft in air space
1
0.000000
+++++
tot_friendly
counts the total friendly AC in air space
1
0.000000
+++++
tot_HVAA
counts number of HVAA in airspace
1
0.000000
+++++
total_com
Total number of OPERATOR communcations
1
0.000000
+++++
total_curren
Total current
1
0.000000
+++++
total_op
Total number of operator actions for an optr.
3
1
5
0.000000
+++++
track

1
0.000000
+++++
Track_HVAAs
track HVAAs
3
1
300
0.000000
+++++
track_id
Track identifier
1
0.000000
+++++
Track_Num
Track number associated with tag
3
1
250
```

```
0.000000
+++++
trackin

1
0.000000
+++++
trackin1
IDS trackin
2
0.000000
+++++
trackin2

1
0.000000
+++++
trackin3

1
0.000000
+++++
tracknum
track num from array
1
0.000000
+++++
trackop

1
0.000000
+++++
tracks

1
0.000000
+++++
tracktest
snapshot debug variable
1
0.000000
+++++
tracktot

2
0.000000
+++++
updt

2
0.000000
+++++
updtop

2
0.000000
+++++
user

1
0.000000
+++++
vector
pursuit=1, nose=2, side=3
1
0.000000
+++++
Vector_Op
pursuit=1, nose=2, side=3 by operator
3
```

```
1
50
0.000000
+++++
watch
number of watchstations
1
0.000000
+++++
watchtot
total num watch stations added in scenario
1
0.000000
+++++
WD
Constant identifier for WD
1
1.000000
+++++
WD_prio
WD priority for final threat determination
3
1
300
0.000000
+++++
WD_req

3
1
300
0.000000
+++++
Weapons
Array - number of weapon types
3
1
250
0.000000
+++++
weapons_ty

1
0.000000
+++++
wpnsin
Number of weapons type for a new track
1
0.000000
+++++
x
x coordinate
2
0.000000
+++++
X
x-coordinate of tag
4
1
500
0.000000
+++++
xAWACS
x-coordinates of AWACS
2
0.000000
+++++
xHVAA
x-coordinates of currently selected HVAA
2
```

```
0.000000
+++++
xin

4
1
50
0.000000
+++++
xop

2
0.000000
+++++
Xpos
x coord of tag and position n
4
2
500
50
0.000000
+++++
xy_distance
distance between aircraft in xy plane
2
0.000000
+++++
Y
y-coordinate of tag
4
1
500
0.000000
+++++
y
y coordinate
2
0.000000
+++++
yAWACS
y-coordinates of AWACS
2
0.000000
+++++
yHVAA
y-coordinates of currently selected HVAA
2
0.000000
+++++
yin

4
1
50
0.000000
+++++
yop

2
0.000000
+++++
Ypos
Y coord of tag and position n
4
2
500
50
0.000000
+++++
zoff
```

```
1
0.000000
+++++
zpos

2
0.000000
+++++
********
ALT_UPDATE
Updates AC Alt from scenario event
Altitude[tagin] := altitudein;

squadin:=(-9999);
altitudein:=(-9999);

---
+++++
CHECK_COURSE
Has next point been reached in pattern flight
time_step := clock - Last_Upd[current]; {in minutes}
n := next_point[current]; {next course change position}
t_to_point := distance(X[current], Y[current], Xpos[current, n], Ypos[current, n]) / (Speed[current] / 60);

if t_to_point < time_step
then Last_Upd[current]+=t_to_point,
     REPOSITION,
     UPDATE_CRS,
     Last_Upd[current]-=(time_step-t_to_point),
     REPOSITION
else
     REPOSITION;



---
+++++
CIE_DATA
Sets up the data for the CIE outpuf file
if not_nearest == 0 & not_fuel1 == 0
then cie_hvaa := 1;
if not_nearest == 0 & not_fuel1 == 1
then cie_hvaa := 2,
if not_nearest == 1 & not_fuel1 == 0
then cie_hvaa := 3;
if not_nearest == 1 & not_fuel1 == 1
then cie_hvaa := 4;

if fuel_ok then cie_fuel2:=1 else cie_fuel2:=0;

if Miss_type1[nearest] then cie_arm:=2;
if Miss_type2[nearest] then cie_arm:=1;
if Miss_type1[nearest] & Miss_type2[nearest] then cie_arm:=3;
if Miss_type1[nearest]==0 & Miss_type2[nearest]==0 then cie_arm:=4;

cie_vector := Vector_Op[index];

---
+++++
CLASS_MATCH
time to classify item
.00045 60,
---
+++++
COMMIT_COUNT
Count the number of commits
{if OKey[WD,i]==1 then WD_commit+=1;}
---
+++++
COMP_HVAA
Compare HVAA committed by exp vs sim op
hvaa_op := FIND_SQUAD;

if hvaa_op <> nearest
then nearest := hvaa_op,
      not_nearest:=1;


---
+++++
CONVERT_TIME
convert to decimal minutes
timein:=((minin*60)+secsin)/60;
---
+++++
DECIDE
time to make decision
.070/60;
---
+++++
DET_HOST
Determine if act is hostile toward friend
{Hostile act is currently based on distance.  Any AC not friendly within 150 of any friendly is considered hostile}

current := Friendly[count];

if occup[current] == 0 &  GET_RANGE < 200
then not_hostile:=0,  hostile_act[tag] := 1;

{checks of hostile heading 'HOST_HEAD' and inside the adid zone 'IN_ADID_ZONE' are also available}

---
+++++
DET_NEAREST
Find the nearest HVAA to target aircraft
count:=1;
nearest:=0,
min_range:=0;

while count <= tot_HVAA do
  DET_OCCUP,
  count+=1;

---
+++++
DET_OCCUP
Determine if HVAA is already occupied
current := HVAA[count];

if occup[current] == 0 then
  DET_RANGE,

---
+++++
DET_RANGE
Determine the range from HVAA to target
xy_distance := distance(X[tag], Y[tag], X[current], Y[current]);  {this is the two dimensional distance between the aircraft in miles}

range := xy_distance,

delta_alt := (absolute(Altitude[tag] - Altitude[current])) / 5280; {altitude delta converted to miles}

{range := ((xy_distance^2) + (delta_alt^2)) ^ (1/2);}
{range is a squared plus b squared equals c squared where a is xy distance, b is altitude difference, and c is range}

if range < min_range | min_range==0 then min_range := range, nearest := current;
---
+++++
DISPATCH
```

```
Dispatch HVAA(s) to hostile AC
Squad_count[Squad_Id[nearest]] -= 2;
if Squad_count[Squad_Id[nearest]] < 2 then occup[nearest]:=1, avail_HVAA-=1;
---
+++++
EYE_FIX
time spent in eye fixation
.250/60;
---
+++++
EYE_MOVE
time spent moving eyes
.170/60;
---
+++++
FIND_SQUAD
find the tag of the squad comitted by sim op
not_done := 1;
count := 1;

while not_done do
  if Com_Hvaa_Op[commit_count+1] == Squad_Id[count]
  then not_done:=0
  else count+=1;

count;

---
+++++
FUEL_CHECK1
Check if HVAA has enough fuel for mission
{Fuel determination is based on the time spent in the air or time since last refueling}

if (clock - Time_In[nearest]) < 30 then fuel_ok:=1
else fuel_ok:=0;

if do_compare & fuel_ok==0
then not_fuel1:=1, fuel_ok:=1;


---
+++++
FUEL_CHECK2
Check if HVAA has fuel vector type
{Fuel determination is based on the time spent in the air or time since last refueling}

if (clock - Time_In[nearest]) < 15
then fuel_ok:=1 else fuel_ok:=0;
---
+++++
GET_BEARING
Get the bearing from on AC to another
direction(X[from], Y[from], X[to], Y[to]);
---
+++++
GET_RANGE
Get the range between two AC
distance(X[current], Y[current], X[tag], Y[tag]);

{current is the friendly being checked against the hostile 'tag'}
---
+++++
HOST_HEAD
Determine if heading is hostile
{Heading is hostile if it is towards a friendly within two degrees}

from := tag;        to := current;
bearing1 := GET_BEARING;

from := current;   to := tag;
```

```
bearing2 := GET_BEARING;

if bearing2+180 > 360 then
  bearing2 := bearing2 + 180 - 360;

if bearing1+2 > 360 then bracket1 := bearing1+2-360;
if bearing1-2 < 0 then bracket2 := bearing1-2+360;

if bearing2 > bracket1 | bearing2 < bracket2
then 0  else 1;

---
+++++
HOSTILE_ACT
Check if AC is making a hostile move
{If turned towards friendly and within specified range or within Air Defense ID Zone}

count:=0;
not_hostile:=1;

while not_hostile & count <= tot_friendly do
  DET_HOST,
  count+=1;

---
+++++
ID_UPDATE
Updates the AC ID based on scenario event
ac_id[tagin] := ID;

squadin:=(-9999);
ID:=(-9999);

---
+++++
IN_ADID_ZONE
Check if AC is in Air Defense ID Zone
x:= X[tag];  y:=Y[tag];

if y>300 | x<(-120) | y<(-100) | x>180
then 0
else if (y-(-1.3333*x)) < (-100) |
        (y-(-2*x)) < 540 |
        (y-(1.125*x)) < (-167.5)
        then 0
        else 1;

---
+++++
LISTEN
time to receive acknowledgement
3.2/60,
---
+++++
MOVE_CUR_T
time to move cursor w/T-ball
{assume approx. 6 cm ave dist to move}
(.3322*(log((6/.25)+.5)))/60;
---
+++++
MOVE_HAND
time to move hand
{assume distance=12cm, size=.25cm}
 0.5586585448239/60;
---
+++++
NEW_COURSE
Used for scenario event course changes
current := tagin;
```

```
REPOSITION;

Pattern[tagin] := pattern;
Positions[tagin] := n+1;

Xpos[tagin, 1] := X[tagin];
Ypos[tagin, 1] := Y[tagin];

count:=1;
while count <= n do
  Xpos[tagin, count+1] := xin[count],
  Ypos[tagin, count+1] := yin[count],
  count+=1;

prev_point[tagin] := 1;
next_point[tagin] := 2;

Course[tagin] := direction(X[tagin], Y[tagin], xin[1], yin[1]);


count:=1;
while count<=n do
  xin[count]:=(-9999),
  yin[count]:=(-9999),
  count+=1;

pattern:=-9999;
n:=-9999;
---
+++++
NEW_TRACK
Start a new track
{Start a new track}

Time_In[tagin]:=timein;
Last_Upd[tagin]:=clock;

Squad_Id[tagin]:=squadin;
Squad_count[squadin]:=squad_num;
Speed[tagin]:=speedin;
Altitude[tagin]:=altitudein;

Positions[tagin]:=n;
Pattern[tagin]:=pattern;

count:=1;
while count <= n do
  Xpos[tagin, count] := xin[count],
  Ypos[tagin, count] := yin[count],
  count+=1;

X[tagin] := xin[1]; Y[tagin] := yin[1];
Course[tagin] := direction(xin[1], yin[1], xin[2], yin[2]);
prev_point[tagin] :=1; next_point[tagin]:=2;

Miss_type1[tagin] :=miss_ty_1;
Miss_type2[tagin] :=miss_ty_2;

ac_id[tagin] :=ID;

start(116,tagin);

tot_aircraft+=1;
aircraft[tot_aircraft] := tagin;

{Re-set input variable values}
timein:=(-9999);
squadin:=(-9999);
squad_num:=(-9999);
speedin:=(-9999);
```

```
altitudein:=(-9999);

count:=1;
while count<=n do
  xin[count]:=(-9999),
  yin[count]:=(-9999),
  count+=1;

n:=(-9999);
pattern:=(-9999);
miss_ty_1:=(-9999);
miss_ty_2:=(-9999);
ID:=(-9999);

---
+++++
PERCEIVE
time to perceive
.100/60;
---
+++++
PUSH_BUTTON
time to push a button
.4/60;
---
+++++
REPOSITION
Reposition individual track
{Determine new position on x,y grid}

{if Course[current]>=0 & Course[current]<90 then }

X[current]:= X[current]+( Speed[current] * ((clock -  Last_Upd[current])/60)*cosine(Course[current]));

 Y[current]:= Y[current]+( Speed[current] *  ((clock -  Last_Upd[current])/60)*sine(Course[current]));


!if Course[current]>=90 & Course[current]<180 then
 X[current]:= X[current]+( Speed[current]*((clock -  Last_Upd[current])/60)*cosine(Course[current]-90)),
 Y[current]:= Y[current]-( Speed[current]*((clock -  Last_Upd[current])/60)*sine(Course[current]-90)), temp:=2;

if Course[current]>=180 & Course[current]<270 then
 X[current]:= X[current]-( Speed[current]*((clock -  Last_Upd[current])/60)*sine(Course[current]-180)),
 Y[current]:= Y[current]-( Speed[current]*((clock -  Last_Upd[current])/60)*cosine(Course[current]-180)), temp:=3;

if Course[current]>=270 & Course[current]<360 then
 X[current]:= X[current]-( Speed[current]*((clock -  Last_Upd[current])/60)*cosine(Course[current]-270)),
 Y[current]:= Y[current]+( Speed[current]*((clock -  Last_Upd[current])/60)*sine(Course[current]-270)), temp:=4;}

Last_Upd[current] := clock;

---
+++++
SPD_UPDATE
Updates the speed from a scenario event
Speed[tagin] := speedin;

speedin:=(-9999);
squadin:=(-9999);
---
+++++
SPEAK
time to issue commands
{assume 6.8 words/transmission}
2/60;
---
+++++
TRACE
Gathers data for the expert trace
task_num := task();
```

```
start(500, 999);
---
+++++
UNNAMED39

---
+++++
UNNAMED42

---
+++++
UPDATE_CRS
route to type 0 or type 1 pattern updates
if Pattern[current] == 0 then UPDATE_CRS0;
if Pattern[current] == 1 then UPDATE_CRS1;
---
+++++
UPDATE_CRS0
Updates course for pattern type 0
if next_point[current] == Positions[current]
then Pattern[current] := 3;

if next_point[current] < Positions[current]
then prev_point[current] := next_point[current],
      next_point[current] += 1,
      n := next_point[current],
      Course[current] := direction(X[current], Y[current], Xpos[current, n], Ypos[current, n]);


---
+++++
UPDATE_CRS1
Updates course for pattern type 1
not_done:=1;

if next_point[current] == Positions[current] & Positions[current]==2 then Pattern[current] := 4,
      time_step := 0,
      t_to_point := 0;

if next_point[current] < Positions[current]
then prev_point[current] := next_point[current],
      next_point[current] += 1,
      n := next_point[current],
Course[current] := direction(X[current], Y[current], Xpos[current, n], Ypos[current, n]), not_done:=0;

if not_done & next_point[current] == Positions[current] & Positions[current]>2
then prev_point[current] := next_point[current],
      next_point[current] := 2, . .
      n := next_point[current],
      Course[current] := direction(X[current], Y[current], Xpos[current, n], Ypos[current, n]);


---
+++++
UPDATE_POS
Update the position of an AC
{AC with pattern type 4 are holding station by circling a point. For the purpose of the model, these will not change position so we will use
REPOSITION with a speed of zero}

if Pattern[current]==4
then temp =Speed[current],
      Speed[current]=0,
      REPOSITION,
      Speed[current]=temp;


{AC with pattern type 3 just do an update without any check for course change}

if Pattern[current]==3 then REPOSITION;
```

```
{AC with pattern type 0 use next point for course until they have reached their last point.  They are then type 3s}

if Pattern[current]==0
then CHECK_COURSE;

{AC with pattern type 1 use next point for course and continue with a cyclical pattern of points}

if Pattern[current]==1
then CHECK_COURSE;
---
+++++
VIS_SEARCH
time spent visually searching
(numfixes*(EYE_MOVE+EYE_FIX))/600;
---
+++++
********
0.000000
3
0
0
1.000000
1
1000.000000
mission_type:=2; {Transition type}
{Only mission type 2 is available at this time}
---
+++++
0.000000
3
0
0
1.000000
1
1000.000000
{Use comparison setting}
{0 = not using  1 = using}
do_compare := 1;
---
+++++
0.000000
3
0
0
1.000000
1
1000.000000
{Simulator Operator Observable Data}
Com_Hvaa_Op[1]:=1018;  Vector_Op[1]:=1;
Com_Hvaa_Op[2]:=1018;  Vector_Op[2]:=1;
Com_Hvaa_Op[3]:=1018;  Vector_Op[3]:=1;
Com_Hvaa_Op[4]:=1018;  Vector_Op[4]:=1;
Com_Hvaa_Op[5]:=1032;  Vector_Op[5]:=1;
Com_Hvaa_Op[6]:=1032;  Vector_Op[6]:=1;
Com_Hvaa_Op[7]:=1018;  Vector_Op[7]:=1;
Com_Hvaa_Op[8]:=1018;  Vector_Op[8]:=1;
Com_Hvaa_Op[9]:=1032;  Vector_Op[5]:=1;
Com_Hvaa_Op[10]:=1032;  Vector_Op[10]:=1;
Com_Hvaa_Op[11]:=1018;  Vector_Op[11]:=1;
Com_Hvaa_Op[12]:=1018;  Vector_Op[12]:=1;
Com_Hvaa_Op[13]:=1018;  Vector_Op[13]:=1;
Com_Hvaa_Op[14]:=1018;  Vector_Op[14]:=1;
Com_Hvaa_Op[15]:=1018;  Vector_Op[15]:=3;
Com_Hvaa_Op[16]:=1018;  Vector_Op[16]:=3;
Com_Hvaa_Op[17]:=1018;  Vector_Op[17]:=3;
Com_Hvaa_Op[18]:=1018;  Vector_Op[18]:=3;
Com_Hvaa_Op[19]:=1034;  Vector_Op[19]:=2;
Com_Hvaa_Op[20]:=1034;  Vector_Op[20]:=2;
Com_Hvaa_Op[21]:=1034;  Vector_Op[21]:=2;
```

```
Com_Hvaa_Op[22]:=1034; Vector_Op[22]:=2;
Com_Hvaa_Op[23]:=1077; Vector_Op[23]:=3;
Com_Hvaa_Op[24]:=1077; Vector_Op[24]:=3;
Com_Hvaa_Op[25]:=1078; Vector_Op[25]:=2;
Com_Hvaa_Op[26]:=1078; Vector_Op[26]:=2;
Com_Hvaa_Op[27]:=1077; Vector_Op[27]:=1;
Com_Hvaa_Op[28]:=1077; Vector_Op[28]:=1;
Com_Hvaa_Op[29]:=1052; Vector_Op[29]:=1;
Com_Hvaa_Op[30]:=1052; Vector_Op[30]:=1;
Com_Hvaa_Op[31]:=1052; Vector_Op[31]:=1;
Com_Hvaa_Op[32]:=1052; Vector_Op[32]:=1;

---
+++++
0.000000
3
0
0
1.000000
1
1000.000000
{Airbase friendly - Jaffna - SLJA}
X[151]:=30; Y[151]:=205;
tot_friendly+=1; Friendly[tot_friendly]:=151;
---
+++++
0.000000
3
0
0
1.000000
1
1000.000000
{Airbase friendly - Trincomallee - SLTE}
X[152] := 65; Y[152] := 135;
tot_friendly+=1; Friendly[tot_friendly] := 152;
---
+++++
0.000000
3
0
0
1.000000
1
1000.000000
{Airbase friendly - Ragooner - SLRR}
X[153] := 35; Y[153] := 115;
tot_friendly+=1; Friendly[tot_friendly] := 153;
---
+++++
0.000000
3
0
0
1.000000
1
1000.000000
{Airbase friendly - Kandy - SLKY}
X[154] := 40; Y[154] := 75;
tot_friendly+=1; Friendly[tot_friendly] := 154;
---
+++++
0.000000
3
0
0
1.000000
1
1000.000000
{Airbase friendly - Colombo - SLCO}
```

```
X[155] := (-5);  Y[155] := 55;
tot_friendly+=1;  Friendly[tot_friendly] := 155;
---
+++++
0.000000
3
0
0
1.000000
1
1000.000000
{Airbase friendly - Galle - SLGE}
X[156] := (23);  Y[156] := 13;
tot_friendly+=1;  Friendly[tot_friendly] := 156;
---
+++++
0.000000
3
0
0
1.000000
1
1000.000000
{Airbase friendly - Hambantota - SLHA}
X[157] := 93;  Y[157] := 28;
tot_friendly+=1;  Friendly[tot_friendly] := 157;
---
+++++
0.000000
3
0
0
1.000000
1
1000.000000
{Carrier}
X[158] := 180;  Y[158] := 180;
tot_friendly+=1;  Friendly[tot_friendly] := 158;
---
+++++
0.000000
3
0
0
1.000000
1
1000.000000
{AWACS HR01}
tagin:=1; timein:=0.00; squadin:=214; squad_num:=1; speedin:=380; altitudein:=29000; n:=5; pattern:=0; xin[1]:=-60; yin[1]:=-60; xin[2]:=60;
yin[2]:=60;xin[3]:=85; yin[3]:=120;xin[4]:=80; yin[4]:=95; xin[5]:=10; yin[5]:=80; miss_ty_1:=0; miss_ty_2:=0; ID:=415; NEW_TRACK;
---
+++++
0.000000
3
0
0
1.000000
1
1000.000000
{DCA MIG-29s BAAZ - Hostile - H000, H001}
tagin:=30; timein:=0.00; squadin:=53; squad_num:=2; speedin:=640; altitudein:=35000; pattern:=1; n:=2; xin[1]:=-110; yin[1]:=328; xin[2]:=-
110; yin[2]:=260; miss_ty_1:=1; miss_ty_1:=2; ID:=9999; NEW_TRACK;

---
+++++
0.000000
3
0
0
1.000000
```

```
1
1000.000000
{DCA F-16Cs - FRAZER 31 - FZ31, 32, 33, 34, 35, 36, 37, 40, 41, 42}
tagin:=13; timein:=0.000; squadin:=1052; squad_num:=10; speedin:=600; altitudein:=32000; pattern:=1; n:=2; xin[1]:=35; yin[1]:=15;
xin[2]:=110; yin[2]:=30; miss_ty_1:=1; miss_ty_2:=1; ID:=9999; NEW_TRACK;


---
+++++
0.000000
3
0
0
1.000000
1
1000.000000
{STRIKE F-111A  - LAZER 30 - LR30, 31, 32, 33, 34, 35}
tagin:=17; timein:=0.00; squadin:=1087; squad_num:=6; speedin:=600; altitudein:=32000; pattern:=1; n:=2; xin[1]:=-5; yin[1]:=55; xin[2]:=110;
yin[2]:=30; miss_ty_1:=0; miss_ty_2:=1; ID:=111; NEW_TRACK;
---
+++++
0.000000
3
0
0
1.000000
1
1000.000000
{PAIL72-Transport}
tagin:=5; timein:=0.0017; squadin:=1005; squad_num:=1; speedin:=500; altitudein:=30000; pattern:=0; n:=2; xin[1]:=-180; yin[1]:=240;
xin[2]:=-5; yin[2]:=55; miss_ty_1:=0; miss_ty_2:=0; ID:=9999; NEW_TRACK;
---
+++++
0.010000
3
0
0
1.000000
1
1000.000000
{Jammers Canberra - Hostile - 3 - H522, H523, 524}
tagin:=134; timein:=0.0083; squadin:=107; squad_num:=1; speedin:=580; altitudein:=28500; pattern:=0; n:=4;  xin[1]:=-135; yin[1]:=245;
xin[2]:=-100; yin[2]:=70; xin[3]:=-10; yin[3]:=65; xin[4]:=-135; yin[4]:=245; miss_ty_1:=0; miss_ty_2:=0; ID:=9999; NEW_TRACK;

tagin:=135; timein:=0.0083; squadin:=108; squad_num:=1; speedin:=580; altitudein:=28500; pattern:=0; n:=4;  xin[1]:=-135; yin[1]:=245;
xin[2]:=-100; yin[2]:=70; xin[3]:=-10; yin[3]:=65; xin[4]:=-135; yin[4]:=245; miss_ty_1:=0; miss_ty_2:=0; ID:=9999; NEW_TRACK;

tagin:=136; timein:=0.0083; squadin:=109; squad_num:=1; speedin:=580; altitudein:=28500; pattern:=0; n:=4;  xin[1]:=-135; yin[1]:=245;
xin[2]:=-100; yin[2]:=70; xin[3]:=-10; yin[3]:=65; xin[4]:=-135; yin[4]:=245; miss_ty_1:=0; miss_ty_2:=0; ID:=9999; NEW_TRACK;
---
+++++
0.020000
3
0
0
1.000000
1
1000.000000
{SOJ EF-111A - Crow 01 - CW01}
tagin:=7; timein:=0.0167; squadin:=1007; squad_num:=1; speedin:=465; altitudein:=24600; n:=3; pattern:=1; xin[1]:=40; yin[1]:=75; xin[2]:=-
60; yin[2]:=180; xin[3]:=-40; yin[3]:=155;
miss_ty_1:=0; miss_ty_2:=0; ID:=111; NEW_TRACK;
---
+++++
0.020000
3
0
0
1.000000
1
```

```
1000.000000
{SL Maritime Surveillance - Cover 01 - CV01}
tagin:=8; timein:=0.0167; squadin:=1008; squad_num:=1; speedin:=245; altitudein:=4000; pattern:=0; n:=10; xin[1]:=-5; yin[1]:=55; xin[2]:=-
120; yin[2]:=60; xin[3]:=-30; yin[3]:=-60; xin[4]:=-20; yin[4]:=-50; xin[5]:=-110; yin[5]:=50; xin[6]:=-100; yin[6]:=50; xin[7]:=-10; yin[7]:=-44;
xin[8]:=0; yin[8]:=-35; xin[9]:=-57; yin[9]:=50; xin[10]:=-5; yin[10]:=55; miss_ty_1:=0; miss_ty_2:=0; ID:=111; NEW_TRACK;


---
+++++
0.020000
3
0
0
1.000000
1
1000.000000
{Air Defense 2 -  Hostile - H0552}
tagin:=68; timein:=0.0167; squadin:=72; squad_num:=1; speedin:=620; altitudein:=27000; pattern:=1; n:=2; xin[1]:=-135; yin[1]:=265; xin[2]:=-
180; yin[2]:=240; miss_ty_1:=1; miss_ty_2:=1; ID:=111; NEW_TRACK;


---
+++++
0.020000
3
0
0
1.000000
1
1000.000000
{DCA F-15C Wolf 51 - WF51, 52, 53, 54, 55, 56, 57, 60, 61, 62, 63, 64, 65, 66}
tagin:=10; timein:=0.0167; squadin:=1018; squad_num:=14; speedin:=640; altitudein:=35000; pattern:=1; n:=2; xin[1]:=65; yin[1]:=135;
xin[2]:=60; yin[2]:=180;  miss_ty_1:=1; miss_ty_2:=1; ID:=111; NEW_TRACK;


---
+++++
0.020000
3
0
0
1.000000
1
1000.000000
{Compass Call EC-130H  Noisy 22 - NY22 }
tagin:=24; timein:=0.0167; squadin:=1112; squad_num:=1; speedin:=276; altitudein:=27500; pattern:=1; n:=4; xin[1]:=-25; yin[1]:=140;
xin[2]:=45; yin[2]:=190; xin[3]:=30; yin[3]:=160; xin[4]:=-25; yin[4]:=140; miss_ty_1:=0; miss_ty_2:=0; ID:=111; NEW_TRACK;


---
+++++
0.020000
3
0
0
1.000000
1
1000.000000
{ABCCC III EC-130E Razorback 31 - RZ31 }
tagin:=26; timein:=0.0167; squadin:=1114; squad_num:=1; speedin:=285; altitudein:=17500; pattern:=1; n:=4; xin[1]:=93; yin[1]:=28; xin[2]:=5;
yin[2]:=125; xin[3]:=45; yin[3]:=135; xin[4]:=27; yin[4]:=129; miss_ty_1:=0; miss_ty_2:=0; ID:=111; NEW_TRACK;


---
+++++
0.020000
3
0
0
1.000000
```

```
1
1000.000000
{Gunship AC-130A Casper 51 - CR51}
tagin:=27; timein:=0.0167; squadin:=1115; squad_num:=1; speedin:=285; altitudein:=14300; pattern:=1; n:=6; xin[1]:=35; yin[1]:=98; xin[2]:=0;
yin[2]:=180; xin[3]:=25; yin[3]:=205; xin[4]:=45; yin[4]:=170; xin[5]:=35; yin[5]:=155; xin[6]:=75; yin[6]:=125; miss_ty_1:=0; miss_ty_2:=0;
ID:=111; NEW_TRACK;


---
+++++
0.020000
3
0
0
1.000000
1
1000.000000
{U-2R ELINT Snoopy 23 - SY26}
tagin:=28; timein:=0.0167; squadin:=1116; squad_num:=1; speedin:=500; altitudein:=70000; pattern:=1; n:=7; xin[1]:=-240; yin[1]:=60;
xin[2]:=-180; yin[2]:=240; xin[3]:=-65; yin[3]:=120; xin[4]:=0; yin[4]:=180; xin[5]:=120; yin[5]:=310; xin[6]:=50; yin[6]:=180; xin[7]:=-65;
yin[7]:=120; miss_ty_1:=0; miss_ty_2:=0; ID:=111; NEW_TRACK;


---
+++++
0.020000
3
0
0
1.000000
1
1000.000000
{DCA F-16Cs - SOLAR - SR11, 12, 13, 14, 15, 16}
tagin:=11; timein:=0.017; squadin:=1032; squad_num:=6; speedin:=600; altitudein:=32000; pattern:=0; n:=6; xin[1]:=23; yin[1]:=13;
xin[2]:=100; yin[2]:=120;  xin[3]:=-10; yin[3]:=180; xin[4]:=-25; yin[4]:=240; xin[5]:=37; yin[5]:=125; xin[6]:=23; yin[6]:=13; miss_ty_1:=1;
miss_ty_2:=1; ID:=111; NEW_TRACK;


---
+++++
0.020000
3
0
0
1.000000
1
1000.000000
{MIG-21 Air Defense F-7M 1 - Hostile - 2 - H050, H054}
tagin:=48; timein:=0.0167; squadin:=62; squad_num:=1; speedin:=550; altitudein:=29000; pattern:=1; n:=2; xin[1]:=-48; yin[1]:=270; xin[2]:=-
25; yin[2]:=240; miss_ty_1:=1; miss_ty_2:=1; ID:=9999; NEW_TRACK;

tagin:=52; timein:=0.0167; squadin:=64; squad_num:=1; speedin:=550; altitudein:=29000; pattern:=1; n:=2; xin[1]:=-48; yin[1]:=270; xin[2]:=-
25; yin[2]:=240; miss_ty_1:=1; miss_ty_2:=1; ID:=111; NEW_TRACK;
---
+++++
0.020000
3
0
0
1.000000
1
1000.000000
{Tanker KC-10 EXXON 40 - EN40 41, 42}
tagin:=29; timein:=0.0167; squadin:=1117; squad_num:=3; speedin:=400; altitudein:=35000; pattern:=1; n:=3; xin[1]:=-28; yin[1]:=115;
xin[2]:=-10; yin[2]:=145; xin[3]:=-20; yin[3]:=90; miss_ty_1:=0; miss_ty_2:=0; ID:=111; NEW_TRACK;


---
+++++
0.020000
```

```
3
0
0
1.000000
1
1000.000000
{Rivet Joint CYPHER 11 - CR11}
tagin:=23; timein:=0.0167; squadin:=1110; squad_num:=1; speedin:=400; altitudein:=31000; pattern:=1; n:=4; xin[1]:=-240; yin[1]:=120;
xin[2]:=-65; yin[2]:=120; xin[3]:=-25; yin[3]:=180; xin[4]:=60; yin[4]:=240;  miss_ty_1:=0; miss_ty_2:=0; ID:=111; NEW_TRACK;


---
+++++
0.030000
3
0
0
1.000000
1
1000.000000
{STRIKE F-15E  - ZIP 01 -  ZP01-07, ZP10-12}
tagin:=15; timein:=0.0333; squadin:=1072; squad_num:=10; speedin:=640; altitudein:=20000; pattern:=1; n:=2; xin[1]:=93; yin[1]:=28; xin[2]:=-
20; yin[2]:=90; miss_ty_1:=0; miss_ty_2:=1; ID:=111; NEW_TRACK;
---
+++++
0.090000
3
0
0
1.000000
1
1000.000000
{STRIKE F-16Ds  - HUSH 10 -  HH10-17, HH20-21}
tagin:=16; timein:=0.0850; squadin:=1077; squad_num:=10; speedin:=600; altitudein:=32000; pattern:=0; n:=6; xin[1]:=23; yin[1]:=13; xin[2]:=-
110, yin[2]:=30; xin[3]:=-10; yin[3]:=180; xin[4]:=-25; yin[4]:=240; xin[5]:=37; yin[5]:=125; xin[6]:=23; yin[6]:=13; miss_ty_1:=0;
miss_ty_2:=1;  ID:=111; NEW_TRACK;


---
+++++
0.170000
3
0
0
1.000000
1
1000.000000
{DCA F-16Cs - SOLAR - SR17, 20, 21, 22}
tagin:=145, timein:=0.1683; squadin:=1034; squad_num:=4; speedin:=600; altitudein:=32000; pattern:=0; n:=6; xin[1]:=23; yin[1]:=13; xin[2]:=-
10, yin[2]:=180; xin[3]:=-25; yin[3]:=240; xin[4]:=-25; yin[4]:=240; xin[5]:=37; yin[5]:=125; xin[6]:=23; yin[6]:=13; miss_ty_1:=1;
miss_ty_2:=1, ID:=111; NEW_TRACK;
---
+++++
0.170000
3
0
0
1.000000
1
1000.000000
{Tanker KC-10 EXXON 40 - EN40 41, 42 - Alt Change}
tagin:=29, timein:=0.1667; squadin:=1117; altitudein:=25500; ALT_UPDATE;
---
+++++
0.170000
3
0
0
1.000000
1
```

```
1000.000000
{DCA F-15E II - EAGLE 01 - EEO1-07, EE10-12}
tagin:=14; timein:=0.1667; squadin:=1062; squad_num:=10; speedin:=500; altitudein:=25000; pattern:=0; n:=5; xin[1]:=40; yin[1]:=75;
xin[2]:=0; yin[2]:=60; xin[3]:=-10; yin[3]:=-95; xin[4]:=-95; yin[4]:=216; xin[5]:=40; yin[5]:=75; miss_ty_1:=0; miss_ty_2:=1; ID:=111;
NEW_TRACK;
---
+++++
0.190000
3
0
0
1.000000
1
1000.000000
{SOJ EF-111A - Crow 01 - CW01 -  ID Change}
tagin:=7; timein:=0.185; squadin:=1007; ID:=211; ID_UPDATE;

---
+++++
0.250000
3
0
0
1.000000
1
1000.000000
{STRIKE F-16Ds  - HUSH 10 - HH22-27, HH30-33}
tagin:=143; timein:=0.2517; squadin:=1078; squad_num:=10; speedin:=600; altitudein:=32000; pattern:=0; n:=5; xin[1]:=23; yin[1]:=13;
xin[2]:=-10; yin[2]:=180; xin[3]:=-25; yin[3]:=240; xin[4]:=37; yin[4]:=125; xin[5]:=23; yin[5]:=13; miss_ty_1:=0; miss_ty_2:=1;  ID:=111;
NEW_TRACK;
---
+++++
0.580000
3
0
0
1.000000
1
1000.000000
{CAS A-10 - Warthog 51 - WG51-56}
tagin:=9; timein:=.5767; squadin:=1009; squad_num:=6; speedin:=250; altitudein:=5000; pattern:=1; n:=2; xin[1]:=35; yin[1]:=115; xin[2]:=5;
yin[2]:=125; miss_ty_1:=0; miss_ty_2:=0; ID:=111; NEW_TRACK;
---
+++++
0.610000
3
0
0
1.000000
1
1000.000000
{DCA MIG-29s BAAZ - Hostile - H000, H001 - Course Change}
tagin:=30; timein:=0.6133; squadin:=53; pattern:=0; n:=4; xin[1]:=-60; yin[1]:=220; xin[2]:=-37; yin[2]:=125; xin[3]:=-110; yin[3]:=260;
xin[4]:=-110; yin[4]:=328; NEW_COURSE;
---
+++++
1.540000
3
0
0
1.000000
1
1000.000000
{STRIKE F-111A  - LAZER 30 - LR30, 31, 32, 33, 34, 35 - Course Change}
tagin:=17; timein:=1.54; squadin:=1087; pattern:=0; n:=3; xin[1]:=-10; yin[1]:=180; xin[2]:=-95; yin[2]:=216;
xin[3]:=-5; yin[3]:=55; NEW_COURSE;
---
+++++
1.840000
3
```

```
0
0
1.000000
1
1000.000000
{Jammers Canberra - Hostile - H524  - New Course}
tagin:=136; timein:=1.8433; squadin:=109; altitudein:=31000; pattern:=0; n:=4;  xin[1]:=-60; yin[1]:=220; xin[2]:=37; yin[2]:=125; xin[3]:=-60;
yin[3]:=220; xin[4]:=-27; yin[4]:=129; NEW_COURSE;
---
+++++
2.630000
3
0
0
1.000000
1
1000.000000
{CAS A-10 - Warthog 51 - WG57, 60, 61, 62}
tagin:=144; timein:= 2.6333; squadin:=1011; squad_num:=4; speedin:=250; altitudein:=5000; pattern:=1; n:=2; xin[1]:=35; yin[1]:=115;
xin[2]:=45; yin[2]:=135; miss_ty_1:=0; miss_ty_2:=0; ID:=111; NEW_TRACK;

---
+++++
3.010000
3
0
0
1.000000
1
1000.000000
{STRIKE F-111A  - LAZER 30 - LR30, 31, 32, 33, 34, 35 - ID Change}
tagin:=17; timein:=3.0147; squadin:=1087; ID:=211; ID_UPDATE;
---
+++++
3.100000
3
0
0
1.000000
1
1000.000000
{Pakastani Maritime Patrol 1212 - H018}
tagin:=6; timein:=3.1033; squadin:=1006; squad_num:=1; speedin:=245; altitudein:=6000; pattern:=0; n:=4; xin[1]:=-117; yin[1]:=120; xin[2]:=-
65; yin[2]:=120; xin[3]:=-57; yin[3]:=50; xin[4]:=-117; yin[4]:=120; miss_ty_1:=0; miss_ty_2:=0; ID:=9999; NEW_TRACK;
---
+++++
3.170000
3
0
0
1.000000
1
1000.000000
{DCA F-16C's - SOLAR - SR11, 12, 13, 14, 15, 16 - ID Change}
tagin:=11; timein:=3.17; squadin:=1032; ID:=211; ID_UPDATE;
---
+++++
3.180000
3
0
0
1.000000
1
1000.000000
{Jammers Canberra - Hostile - H523, 524 - ID Change}
tagin:=135; timein:=3.1767; squadin:=108; ID:=111; ID_UPDATE;
tagin:=136; timein:=3.1783; squadin:=109; ID:=111; ID_UPDATE;
---
+++++
3.180000
```

```
3
0
0
1.000000
1
1000.000000
{Rivet Joint CYPHER 11 - CR11 - ID Change}
tagin:=23; timein:=3.185; squadin:=1110; ID:=211; ID_UPDATE;
---
+++++
3.180000
3
0
0
1.000000
1
1000.000000
{ABCCC III EC-130E Razorback 31 - RZ31 - ID Change}
tagin:=26; timein:=3.185; squadin:=1114; ID:=211; ID_UPDATE;


---
+++++
3.180000
3
0
0
1.000000
1
1000.000000
{Compass Call EC-130H  Noisy 22 - NY22 - ID Change}
tagin:=24; timein:=3.185; squadin:=1112; ID:=211; ID_UPDATE;


---
+++++
3.180000
3
0
0
1.000000
1
1000.000000
{Gunship AC-130A Casper 51 - CR51 - ID Change}
tagin:=27; timein:=03.185; squadin:=1115; ID:=211; ID_UPDATE;
---
+++++
3.200000
3
0
0
1.000000
1
1000.000000
{SI  Maritime Surveillance - Cover 01 - CV01 -  ID Change}
tagin:=8; timein:=3.2017; squadin:=1008; ID:=211; ID_UPDATE;
---
+++++
3.200000
3
0
0
1.000000
1
1000.000000
{U-2R ELINT Snoopy 23 - SY26 - ID Change}
tagin:=28; timein:=3.2017; squadin:=1116; ID:=211; ID_UPDATE;
---
+++++
3.200000
```

```
3
0
0
1.000000
1
1000.000000
{Tanker KC-10 EXXON 40 - EN40 41, 42 - ID Change}
tagin:=29; timein:=3.2017; squadin:=1117; ID:=211; ID_UPDATE;
---
+++++
3.200000
3
0
0
1.000000
1
1000.000000
{STRIKE F-15E  - ZIP 01 -  ZP01-07, ZP10-12 - ID Change}
tagin:=15; timein:=3.2017; squadin:=1072; ID:=211; ID_UPDATE;
---
+++++
3.220000
3
0
0
1.000000
1
1000.000000
{DCA F-15C Wolf 51 - WF51, 52, 53, 54, 55, 56, 57, 60, 61, 62, 63, 64, 65, 66  -  ID Change}
tagin:=10; timein:=3.2183; squadin:=1018; ID:=211; ID_UPDATE;
---
+++++
3.250000
3
0
0
1.000000
1
1000.000000
{STRIKE F-16Ds  - HUSH 10 -  HH10-17, HH20-21 - ID Change}
tagin:=16; timein:=3.2533; squadin:=1077; ID:=211; ID_UPDATE;
---
+++++
3.300000
3
0
0
1.000000
1
1000.000000
{CAS A-10 - Warthog 51 - WG57, 60, 61, 62  -  ID Change}
tagin:=144; timein:=3.3017; squadin:=1011; ID:=211; ID_UPDATE;
---
+++++
3.340000
3
0
0
1.000000
1
1000.000000
{DCA F-15E II - EAGLE 01 - EE01-07, EE10-12 - ID Change}
tagin:=14; timein:=3.3350; squadin:=1062; ID:=211; ID_UPDATE;
---
+++++
3.340000
3
0
0
1.000000
```

```
1
1000.000000
{DCA F-16Cs - SOLAR - SR17, 20, 21, 22 - ID Change}
tagin:=145; timein:=3.335; squadin:=1034; ID:=211; ID_UPDATE;
---
+++++
3.380000
3
0
0
1.000000
1
1000.000000
{MIG-21 Air Defense F-7M 1 - Hostile - H050, H054- ID Change}
tagin:=52; timein:=3.385; squadin:=64; ID:=111; ID_UPDATE;
---
+++++
3.420000
3
0
0
1.000000
1
1000.000000
{STRIKE F-16Ds  - HUSH 10 - HH22-27, HH30-33 - ID Change}
tagin:=143; timein:=3.42; squadin:=1078; ID:=211; ID_UPDATE;
---
+++++
3.430000
3
0
0
1.000000
1
1000.000000
{Air Defense 2 -  Hostile - H0552 - ID Change}
tagin:=68; timein:=3.435; squadin:=72; ID:=141; ID_UPDATE;
---
+++++
3.510000
3
0
0
1.000000
1
1000.000000
{Jammers Canberra - Hostile - H524  - ID Change}
tagin:=136; timein:=3.51; squadin:=109; ID:=141; ID_UPDATE;
---
+++++
3.610000
3
0
0
1.000000
1
1000.000000
{Pakastani Maritime Patrol 1212 - H018  -  ID Change}
tagin:=6; timein:=3.6050, squadin:=1006; ID:=111; ID_UPDATE;
---
+++++
3.620000
3
0
0
1.000000
1
1000.000000
{DCA MIG-29s BAAZ - Hostile - H000, H001 - ID Change}
tagin:=30; timein:=3.615; squadin:=53; ID:=111; ID_UPDATE;
```

```
---
+++++
3.670000
3
0
0
1.000000
1
1000.000000
{PAIL72-Transport  - ID Change}
tagin:=5; timein:=3.6683; squadin:=1005; ID:=111; ID_UPDATE;
---
+++++
3.680000
3
0
0
1.000000
1
1000.000000
{Jammers Canberra - Hostile - H524  - ID Change}
tagin:=136; timein:=3.6767; squadin:=109; ID:=311; ID_UPDATE;
---
+++++
3.770000
3
0
0
1.000000
1
1000.000000
{Pakastani Maritime Patrol 1212 - H018  - ID Change}
tagin:=6; timein:=3.77; squadin:=1006; ID:=121; ID_UPDATE;
---
+++++
3.840000
3
0
0
1.000000
1
1000.000000
{Jammers Canberra - Hostile  - H523  - ID Change}
tagin:=135; timein:=3.8417; squadin:=108; ID:=141; ID_UPDATE;
---
+++++
3.910000
3
0
0
1.000000
1
1000.000000
{CAS A-10 - Warthog 51 - WG51-56  - ID Change}
tagin:=9; timein:=3.9117; squadin:=1009; ID:=211;  ID_UPDATE;
---
+++++
3.940000
3
0
0
1.000000
1
1000.000000
{Pakastani Maritime Patrol 1212 - H018  -  ID Change}
tagin:=6; timein:=3.9383; squadin:=1006; ID:=137; ID_UPDATE;

---
+++++
5.230000
```

```
3
0
0
1.000000
1
1000.000000
{Jammers Canberra - Hostile - H522 - Course Change}
tagin:=134; timein:=5.2317; squadin:=107; pattern:=0; n:=4;  xin[1]:=-60; yin[1]:=220; xin[2]:=-30; yin[2]:=205; xin[3]:=-60; yin[3]:=220;
xin[4]:=-135; yin[4]:=245; NEW_COURSE;
---
+++++
5.320000
3
0
0
1.000000
1
1000.000000
{MIG-21 Air Defense F-7M 1 - Hostile - H054- ID Change}
tagin:=52; timein:=5.3167; squadin:=64; ID:=141; ID_UPDATE;
---
+++++
5.400000
3
0
0
1.000000
1
1000.000000
{Jammers Canberra - Hostile - H522 - ID Change}
tagin:=134; timein:=5.4; squadin:=107; ID:=111; ID_UPDATE;
---
+++++
5.570000
3
0
0
1.000000
1
1000.000000
{Jammers Canberra - Hostile - H522 - ID Change}
tagin:=134; timein:=5.565; squadin:=107; ID:=141; ID_UPDATE;
---
+++++
5.670000
3
0
0
1.000000
1
1000.000000
{PAIL72-Transport  -  ID Change}
tagin:=5; timein:=5.6667; squadin:=1005; ID:=211; ID_UPDATE;

---
+++++
5.730000
3
0
0
1.000000
1
1000.000000
{Jammers Canberra - Hostile - H522 - ID Change}
tagin:=134; timein:=5.7317; squadin:=107; ID:=311; ID_UPDATE;
---
+++++
5.780000
3
0
```

```
0
1.000000
1
1000.000000
{DCA MIG-29s BAAZ - Hostile - H000, H001 - ID Change}
tagin:=30; timein:=5.78; squadin:=53; ID:=141; ID_UPDATE;
---
+++++
6.010000
3
0
0
1.000000
1
1000.000000
{Air Defense MIG-23 FLOGGER-B - H0433 - Hostile}
tagin:=60; timein:=6.01; squadin:=68; squad_num:=2; speedin:=580; altitudein:=30000; pattern:=1; n:=2; xin[1]:=-115; yin[1]:=211; xin[2]:=-
125; yin[2]:=180; miss_ty_1:=1; miss_ty_2:=1; ID:=9999; NEW_TRACK;
---
+++++
6.010000
3
0
0
1.000000
1
1000.000000
{DCA F-16Cs - FRAZER 31 - FZ31, 32, 33, 34, 35, 36, 37, 40, 41, 42 - Course Change}
tagin:=13; timein:=6.01; squadin:=1052;  pattern:=0; n:=5; xin[1]:=110; yin[1]:=30; xin[2]:=-95; yin[2]:=216; xin[3]:=-100; yin[3]:=170;
xin[4]:=37; yin[4]:=125; xin[5]:=35; yin[5]:=115; NEW_COURSE;
---
+++++
6.010000
3
0
0
1.000000
1
1000.000000
{Air Defense MIG-23 FLOGGER-B - H0433 - Hostile - Course Change}
tagin:=60; timein:=6.012; squadin:=68; pattern:=0; n:=4; xin[1]:=-100; yin[1]:=170; xin[2]:=-10; yin[2]:=65; xin[3]:=-100; yin[3]:=170;
xin[4]:=-115; yin[4]:=211; NEW_COURSE;
---
+++++
6.180000
3
0
0
1.000000
1
1000.000000
{Air Defense MIG-23 FLOGGER-B - H0433 - Hostile - ID Change}
tagin:=60; timein:=6.1783; squadin:=68; ID:=111; ID_UPDATE;
---
+++++
6.280000
3
0
0
1.000000
1
1000.000000
{DCA F-16Cs - FRAZER 31 - FZ31, 32, 33, 34, 35, 36, 37, 40, 41, 42 - ID Change}
tagin:=13; timein:=6.28; squadin:=1052; ID:=211; ID_UPDATE;

---
+++++
6.340000
3
0
```

```
0
1.000000
1
1000.000000
{Air Defense MIG-23 FLOGGER-B - H0433 - Hostile - ID Change}
tagin:=60; timein:=6.3433; squadin:=68; ID:=141; ID_UPDATE;
---
+++++
6.490000
3
0
0
1.000000
1
1000.000000
{Ground Attack MIG-27 BAHADUR - H0277 - Hostile}
tagin:=114; timein:=6.495; squadin:=95; squad_num:=1; speedin:=600; altitudein:=29000; pattern:=0; n:=5; xin[1]:=-80; yin[1]:=247; xin[2]:=-
60; yin[2]:=220; xin[3]:=37; yin[3]:=125; xin[4]:=-60; yin[4]:=220; xin[5]:=-80; yin[5]:=247; miss_ty_1:=0; miss_ty_2:=1; ID:=9999;
NEW_TRACK;
---
+++++
6.510000
3
0
0
1.000000
1
1000.000000
{Air Defense MIG-23 FLOGGER-B - H0433 - Hostile - ID Change}
tagin:=60; timein:=6.51; squadin:=68; ID:=311; ID_UPDATE;
---
+++++
6.630000
3
0
0
1.000000
1
1000.000000
{Ground Attack MIG-27 BAHADUR - H0277 - Hostile - ID Change}
tagin:=114; timein:=6.633; squadin:=95; ID:=111; ID_UPDATE;
---
+++++
6.830000
3
0
0
1.000000
1
1000.000000
{Ground Attack MIG-27 BAHADUR - H0277 - Hostile - ID Change}
tagin:=114; timein:=6.8283; squadin:=95; ID:=141; ID_UPDATE;
---
+++++
6.910000
3
0
0
1.000000
1
1000.000000
{Ground Attack MIG-27 BAHADUR - H0301 - Hostile}
tagin:=116; timein:=6.9116; squadin:=96; squad_num:=1; speedin:=600; altitudein:=29000; pattern:=0; n:=5; xin[1]:=-80; yin[1]:=247; xin[2]:=-
60; yin[2]:=220; xin[3]:=37; yin[3]:=125; xin[4]:=-100; yin[4]:=170; xin[5]:=-80; yin[5]:=247; miss_ty_1:=0; miss_ty_2:=1; ID:=9999;
NEW_TRACK;
---
+++++
7.000000
3
0
```

```
0
1.000000
1
1000.000000
{Ground Attack MIG-27 BAHADUR - H0277 - Hostile - ID Change}
tagin:=114; timein:=7.000; squadin:=95; ID:=311; ID_UPDATE;
---
+++++
7.240000
3
0
0
1.000000
1
1000.000000
{Ground Attack MIG-27 BAHADUR - H0301 - Hostile - ID Change}
tagin:=116; timein:=7.245; squadin:=96; ID:=141; ID_UPDATE;
---
+++++
7.380000
3
0
0
1.000000
1
1000.000000
{MIG-21 Air Defense F-7M 1 - Hostile - H054- ID Change}
tagin:=52; timein:=7.3833; squadin:=64; ID:=311; ID_UPDATE;
---
+++++
7.420000
3
0
0
1.000000
1
1000.000000
{Ground Attack MIG-27 BAHADUR - H0301 - Hostile - ID Change}
tagin:=116; timein:=7.4167; squadin:=96; ID:=311; ID_UPDATE;
---
+++++
7.430000
3
0
0
1.000000
1
1000.000000
{Air Defense 2 -  Hostile - H0552 - ID Change}
tagin:=68; timein:=7.4333; squadin:=72; ID:=311; ID_UPDATE;
---
+++++
7.780000
3
0
0
1.000000
1
1000.000000
{DCA MIG-29s BAAZ - Hostile - H000, H001 - ID Change}
tagin:=30; timein:=7.78; squadin:=53; ID:=311; ID_UPDATE;
---
+++++
8.110000
3
0
0
1.000000
1
1000.000000
```

```
{Air Defense MIG-23 FLOGGER-B - H0431 - Hostile}
tagin:=58; timein:=8.1117; squadin:=67; squad_num:=1; speedin:=580; altitudein:=30000; pattern:=1; n:=2; xin[1]:=-115; yin[1]:=211; xin[2]:=-
125; yin[2]:=180; miss_ty_1:=1; miss_ty_2:=1; ID:=9999; NEW_TRACK;
---
+++++
8.120000
3
0
0
1.000000
1
1000.000000
{Air Defense MIG-23 FLOGGER-B - H0431 - Hostile - Course Change}
tagin:=58; timein:=8.12; squadin:=67; pattern:=0; n:=4; xin[1]:=-60; yin[1]:=220; xin[2]:=30; yin[2]:=205; xin[3]:=-60; yin[3]:=220; xin[4]:=-
115; yin[4]:=211; NEW_COURSE;
---
+++++
8.190000
3
0
0
1.000000
1
1000.000000
{Air Defense MIG-23 FLOGGER-B - H0431 - Hostile - ID Change}
tagin:=58; timein:=8.1933; squadin:=67; ID:=111; ID_UPDATE;
---
+++++
8.360000
3
0
0
1.000000
1
1000.000000
{Air Defense MIG-23 FLOGGER-B - H0431 - Hostile - ID Change}
tagin:=58; timein:=8.3583; squadin:=67; ID:=121; ID_UPDATE;
---
+++++
8.480000
3
0
0
1.000000
1
1000.000000
{CAS A-10 - Warthog 51 - WG57, 60, 61, 62  -  Course Change}
tagin:=144; timein:=8.4767; squadin:=1011; pattern:=0; n:=2; xin[1]:=75; yin[1]:=125; xin[2]:=35; yin[2]:=115; NEW_COURSE;
---
+++++
8.510000
3
0
0
1.000000
1
1000.000000
{CAS A-10 - Warthog 51 - WG51-56  -  Course Change}
tagin:=9; timein:=8.51; squadin:=1009; pattern:=0; n:=3; xin[1]:=5; yin[1]:=125; xin[2]:=30; yin[2]:=160; xin[3]:=35; yin[3]:=115;
NEW_COURSE;
---
+++++
8.570000
3
0
0
1.000000
1
1000.000000
{Jammers Canberra - Hostile - H523 - Alt & Spd Change}
```

```
tagin:=135; timein:=8.5717; squadin:=108; speedin:=500; altitudein:=5000; ALT_UPDATE;  SPD_UPDATE;
---
+++++
9.170000
3
0
0
1.000000
1
1000.000000
{Air Defense MIG-23 FLOGGER-B - H0431 - Hostile - ID Change}
tagin:=58; timein:=9.1667; squadin:=67; ID:=141; ID_UPDATE;
---
+++++
9.330000
3
0
0
1.000000
1
1000.000000
{Air Defense MIG-23 FLOGGER-B - H0431 - Hostile - ID Change}
tagin:=58; timein:=9.3333; squadin:=67; ID:=311; ID_UPDATE;
---
+++++
9.850000
3
0
0
1.000000
1
1000.000000
{Ground Attack MIG-27 BAHADUR - H0277 - Hostile - Spd Change}
tagin:=114; timein:=9.855; squadin:=95; speedin:=550; SPD_UPDATE;
---
+++++
10.020000
3
0
0
1.000000
1
1000.000000
{Jammers Canberra - Hostile - H524 - Spd Change}
tagin:=136; timein:=10.0217; squadin:=109; speedin:=550; SPD_UPDATE;
---
+++++
10.270000
3
0
0
1.000000
1
1000.000000
{Ground Attack MIG-27 BAHADUR - H0301 - Hostile - ID Change}
tagin:=116; timein:=10.2717; squadin:=96; speedin:=550; SPD_UPDATE;
---
+++++
10.460000
3
0
0
1.000000
1
1000.000000
{MIG-21 Air Defense F-7M 1 - Hostile - H054- Course Change}
tagin:=48; timein:=10.4633; squadin:=62; pattern:=0; n:=4; xin[1]:=-60; yin[1]:=220; xin[2]:=30; yin[2]:=205; xin[3]:=-60; yin[3]:=220;
xin[4]:=-48; yin[4]:=270; NEW_COURSE;
---
+++++
10.500000
3
0
0
1.000000
1
1000.000000
{MIG-21 Air Defense F-7M 1 - Hostile - H050- ID Change}
tagin:=48; timein:=10.5033; squadin:=62; ID:=111; ID_UPDATE;
---
+++++
10.670000
3
0
0
1.000000
1
1000.000000
{MIG-21 Air Defense F-7M 1 - Hostile - H050- ID Change}
tagin:=48; timein:=10.6667; squadin:=62; ID:=121; ID_UPDATE;
---
+++++
10.900000
3
0
0
1.000000
1
1000.000000
{STRIKE F-15E  - ZIP 01 -  ZP01-07, ZP10-12 - Spd Change}
tagin:=15; timein:=10.905; squadin:=1072; speedin:=450; SPD_UPDATE;
---
+++++
10.930000
3
0
0
1.000000
1
1000.000000
{Ground Attack MIG-23H FLOGGER-H - H0347 - Hostile}
tagin:=124; timein:=10.9283; squadin:=100; squad_num:=1; speedin:=580; altitudein:=28500; pattern:=0; n:=5; xin[1]:=-480; yin[1]:=270;
xin[2]:=-60; yin[2]:=220; xin[3]:=30; yin[3]:=205; xin[4]:=-60; yin[4]:=220; xin[5]:=-480; yin[5]:=270; miss_ty_1:=0; miss_ty_2:=1; ID:=9999;
NEW_TRACK;
---
+++++
11.000000
3
0
0
1.000000
1
1000.000000
{MIG-21 Air Defense F-7M 1 - Hostile - H050- ID Change}
tagin:=48; timein:=11.00; squadin:=62; ID:=141; ID_UPDATE;
---
+++++
11.040000
3
0
0
1.000000
1
1000.000000
{Ground Attack MIG-23H FLOGGER-H - H0347 - Hostile - ID Change}
tagin:=124; timein:=11.0367; squadin:=100; ID:=111; ID_UPDATE;
---
+++++
11.470000
3
```

```
0
0
1.000000
1
1000.000000
{Ground Attack MIG-23H FLOGGER-H - H0347 - Hostile - ID Change}
tagin:=124; timein:=11.4733; squadin:=100; ID:=121; ID_UPDATE;
---
+++++
11.500000
3
0
0
1.000000
1
1000.000000
{MIG-21 Air Defense F-7M 1 - Hostile - H050- ID Change}
tagin:=48; timein:=11.50; squadin:=62; ID:=311; ID_UPDATE;
---
+++++
12.040000
3
0
0
1.000000
1
1000.000000
{Ground Attack MIG-23H FLOGGER-H - H0347 - Hostile - ID Change}
tagin:=124; timein:=12.04; squadin:=100; ID:=141; ID_UPDATE;
---
+++++
13.010000
3
0
0
1.000000
1
1000.000000
{Jammers Canberra - Hostile - H522 - Alt & Spd Change}
tagin:=134; timein:=13.0083; squadin:=107; altitudein:=15000; speedin:=500; ALT_UPDATE; SPD_UPDATE;
---
+++++
13.010000
3
0
0
1.000000
1
1000.000000
{MIG-21 Air Defense F-7M 1 - Hostile - H050- Alt Change}
tagin:=48; timein:=13.01; squadin:=62; altitudein:=16000; speedin:=500; ALT_UPDATE; SPD_UPDATE;
---
+++++
13.330000
3
0
0
1.000000
1
1000.000000
{STRIKE F-15E  - ZIP 01 -  ZP01-07, ZP10-12 - Course Change}
tagin:=15; timein:=13.3333; squadin:=1072; speedin:=720; altitudein:=30000; pattern:=0; n:=4; xin[1]:=-10; yin[1]:=180; xin[2]:=-100;
yin[2]:=170; xin[3]:=37; yin[3]:=125; xin[4]:=93; yin[4]:=28; NEW_COURSE;
---
+++++
13.340000
3
0
0
1.000000
```

```
1
1000.000000
{Ground Attack MIG-23H FLOGGER-H - H0347 - Hostile - Alt & Spd Change}
tagin:=124; timein:=13.3417; squadin:=100; altitudein:=1000; speedin:=500; ALT_UPDATE; SPD_UPDATE;
---
+++++
13.360000
3
0
0
1.000000
1
1000.000000
{Ground Attack MIG-23H FLOGGER-H - H0347 - Hostile - ID Change}
tagin:=124; timein:=13.3583; squadin:=100; ID:=311; ID_UPDATE;
tagin:=124; timein:=13.3583; squadin:=100; speedin:=580; altitudein:=28500; ALT_UPDATE; SPD_UPDATE;
---
+++++
13.540000
3
0
0
1.000000
1
1000.000000
{Air Defense 3 - H0576 - Hostile}
tagin:=78; timein:=13.5417; squadin:=77; squad_num:=1; speedin:=450; altitudein:=5000; pattern:=0; n:=3; xin[1]:=-180; yin[1]:=60; xin[2]:=-
120; yin[2]:=60; xin[3]:=-180; yin[3]:=60;  miss_ty_1:=1; miss_ty_2:=1; ID:=9999; NEW_TRACK;
---
+++++
13.610000
3
0
0
1.000000
1
1000.000000
{Ground Attack JAGUAR - H0263 - Hostile}
tagin:=112; timein:=13.6133; squadin:=94; squad_num:=1; speedin:=600; altitudein:=25000; pattern:=0; n:=5; xin[1]:=-110; yin[1]:=238;
xin[2]:=-60; yin[2]:=120; xin[3]:=-30; yin[3]:=205; xin[4]:=-60; yin[4]:=120; xin[5]:=-85; yin[5]:=252;  miss_ty_1:=0; miss_ty_2:=0; ID:=9999;
NEW_TRACK.
---
+++++
13.670000
3
0
0
1.000000
1
1000.000000
{Air Defense 3 - H0576 - Hostile - ID Change}
tagin:=78; timein:=13.670; squadin:=77; ID:=111; ID_UPDATE;
---
+++++
13.860000
3
0
0
1.000000
1
1000.000000
{Air Defense 3 - H0576 - Hostile - ID Change}
tagin:=78; timein:=13.8583; squadin:=77; ID:=121; ID_UPDATE;
---
+++++
14.040000
3
0
0
1.000000
```

```
1
1000.000000
{Air Defense 3 - H0576 - Hostile - ID Change}
tagin:=78; timein:=14.0417; squadin:=77; ID:=141; ID_UPDATE;
---
+++++
14.050000
3
0
0
1.000000
1
1000.000000
{STRIKE F-111A  - LAZER 30 - LR30, 31, 32, 33, 34, 35 - Alt Change}
tagin:=17; timein:=14.05; squadin:=1087; speedin:=580; altitudein:=500; ALT_UPDATE; SPD_UPDATE;
---
+++++
14.120000
3
0
0
1.000000
1
1000.000000
{DCA F-15E II - EAGLE 01 - EEO1-07, EE10-12 - Alt Change}
tagin:=14; timein:=14.1217; squadin:=1062; altitudein:=1000; ALT_UPDATE;
---
+++++
14.230000
3
0
0
1.000000
1
1000.000000
{Air Defense 3 - H0576 - Hostile - ID Change}
tagin:=78; timein:=14.225; squadin:=77; ID:=311; ID_UPDATE;
---
+++++
14.670000
3
0
0
1.000000
1
1000.000000
{Ground Attack JAGUAR - H0263 - Hostile - ID Change}
tagin:=112; timein:=14.6683; squadin:=94; ID:=311; ID_UPDATE;

---
+++++
15.090000
3
0
0
1.000000
1
1000.000000
{Air Defense MIG-23 FLOGGER-B - H0431 - Hostile - Alt & Spd Change}
tagin:=58; timein:=15.09; squadin:=67; speedin:=620; altitudein:=25000; ALT_UPDATE; SPD_UPDATE;
---
+++++
16.330000
3
0
0
1.000000
1
1000.000000
{CAS A-10 - Warthog 51 - WG57, 60, 61, 62  - Alt & Spd Change}
```

```
tagin:=144; timein:=16.3333; squadin:=1011; speedin:=250; altitudein:=5000; ALT_UPDATE; SPD_UPDATE;
---
+++++
17.020000
3
0
0
1.000000
1
1000.000000
{DCA F-16Cs - SOLAR - SR11, 12, 13, 14, 15, 16 - Alt Change}
tagin:=11; timein:=17.025; squadin:=1032; altitudein:=500; ALT_UPDATE;
---
+++++
17.030001
3
0
0
1.000000
1
1000.000000
{Ground Attack MIG-27 BAHADUR - H0277 - Hostile - Alt Change}
tagin:=114; timein:=17.03; squadin:=95; altitudein:=500; ALT_UPDATE;
---
+++++
17.110001
3
0
0
1.000000
1
1000.000000
{STRIKE F-16Ds - HUSH 10 - HH10-17, HH20-21 - ID Change}
tagin:=16; timein:=17.1083; squadin:=1077; altitudein:=500; ALT_UPDATE;
---
+++++
17.190001
3
0
0
1.000000
1
1000.000000
{DCA F-16Cs - SOLAR - SR17, 20, 21, 22 - Alt Change}
tagin:=145; timein:=17.1917; squadin:=1034; altitudein:=500; ALT_UPDATE;
---
+++++
17.200001
3
0
0
1.000000
1
1000.000000
{Jammers Canberra - Hostile - H524 - Alt Change}
tagin:=136; timein:=17.1967; squadin:=109; altitudein:=500; ALT_UPDATE;
---
+++++
17.270000
3
0
0
1.000000
1
1000.000000
{STRIKE F-16Ds - HUSH 10 - HH22-27, HH30-33 - ID Change}
tagin:=143; timein:=17.275; squadin:=1078; altitudein:=500; ALT_UPDATE;
---
+++++
17.450001
```

```
3
0
0
1.000000
1
1000.000000
{Ground Attack MIG-27 BAHADUR - H0301 - Hostile - Alt Change}
tagin:=116; timein:=17.4467; squadin:=96; altitudein:=500; ALT_UPDATE;
---
+++++
17.940001
3
0
0
1.000000
1
1000.000000
{Air Defense MIG-23 FLOGGER-B- H0433 - Hostile - Alt Change}
tagin:=60; timein:=17.9367; squadin:=68; altitudein:=1000; ALT_UPDATE;
---
+++++
17.940001
3
0
0
1.000000
1
1000.000000
{DCA MIG-29s BAAZ - Hostile - H000, H001 - Alt Change}
tagin:=30; timein:=17.9367; squadin:=53; altitudein:=1000; ALT_UPDATE;
---
+++++
18.170000
3
0
0
1.000000
1
1000.000000
{CAS A-10 - Warthog 51 - WG51-56  - Alt & Spd Change}
tagin:=9; timein:=18.1667; squadin:=1009; speedin:=200; altitudein:=500; ALT_UPDATE; SPD_UPDATE;
---
+++++
19.000000
3
0
0
1.000000
1
1000.000000
{CAS A-10 - Warthog 51 - WG51-56  - Alt & Spd Change}
tagin:=9; timein:=19.00; squadin:=1009; speedin:=250; altitudein:=5000; ALT_UPDATE; SPD_UPDATE;
---
+++++
21.670000
3
0
0
1.000000
1
1000.000000
{S1 Maritime Surveillance - Cover 01 - CV01  - Alt Change}
tagin:=8; timein:=21.6667; squadin:=1008; altitudein:=6000; ALT_UPDATE;

---
+++++
25.080000
3
0
0
```

```
1.000000
1
1000.000000
{MIG-21 Air Defense F-7M 1 - Hostile - H050- Alt Change}
tagin:=48; timein:=25.085; squadin:=62; speedin:=580; ALT_UPDATE; SPD_UPDATE;

---
+++++
25.090000
3
0
0
1.000000
1
1000.000000
{Jammers Canberra - Hostile - H522 - Spd Change}
tagin:=134; timein:=25.0883; squadin:=107; speedin:=580; SPD_UPDATE;
---
+++++
25.170000
3
0
0
1.000000
1
1000.000000
{Jammers Canberra - Hostile  - H523 - Spd Change}
tagin:=135; timein:=25.1667; squadin:=108; speedin:=580; SPD_UPDATE;
---
+++++
29.100000
3
0
0
1.000000
1
1000.000000
{STRIKE F-111A  - LAZER 30 - LR30, 31, 32, 33, 34, 35 - Alt Change}
tagin:=17; timein:=29.1; squadin:=1087; altitudein:=32000; ALT_UPDATE;
---
+++++
29.690001
3
0
0
1.000000
1
1000.000000
{DCA F-16Cs - FALCON - FN27, 30, 31, 32}
tagin:=12; timein:=29.6950; squadin:=1042; squad_num:=4; speedin:=600; altitudein:=32000; pattern:=0; n:=7; xin[1]:=93; yin[1]:=28;
xin[2]:=0; yin[2]:=60; xin[3]:=-10; yin[3]:=180; xin[4]:=-115; yin[4]:=211; xin[5]:=-25; yin[5]:=240; xin[6]:=-10; yin[6]:=145; xin[7]:=93;
yin[7]:=28; miss_ty_1:=1; miss_ty_2:=1; ID:=9999; NEW_TRACK;
---
+++++
30.530001
3
0
0
1.000000
1
1000.000000
{DCA F-16Cs - FALCON - FN21-26}
tagin:=20; timein:=30.5283; squadin:=1044; squad_num:=6; speedin:=600; altitudein:=32000; pattern:=0; n:=7; xin[1]:=93; yin[1]:=28;
xin[2]:=0; yin[2]:=60; xin[3]:=-10; yin[3]:=180; xin[4]:=-115; yin[4]:=211; xin[5]:=-25; yin[5]:=240; xin[6]:=-10; yin[6]:=145; xin[7]:=93;
yin[7]:=28; miss_ty_1:=1; miss_ty_2:=1; ID:=9999; NEW_TRACK;
---
+++++
31.059999
3
0
```

```
0
1.000000
1
1000.000000
{DCA MIG-29s BAAZ - Hostile - H000, H001 - Alt Change}
tagin:=30; timein:=31.0633; squadin:=53; altitudein:=35000; ALT_UPDATE;
---
+++++
31.559999
3
0
0
1.000000
1
1000.000000
{Air Defense MIG-23 FLOGGER-B - H0433 - Hostile - Alt Change}
tagin:=60; timein:=31.5583; squadin:=68; altitudein:=30000; ALT_UPDATE;
---
+++++
31.700001
3
0
0
1.000000
1
1000.000000
{DCA F-16Cs - FALCON - FN27, 30, 31, 32 - ID Change}
tagin:=12; timein:=31.6967; squadin:=1042; ID:=211; ID_UPDATE;
---
+++++
31.799999
3
0
0
1.000000
1
1000.000000
{DCA F-15E II - EAGLE 01 - EEO1-07, EE10-12 - Alt Change}
tagin:=14; timein:=31.8; squadin:=1062; altitudein:=25000; ALT_UPDATE;

---
+++++
31.889999
3
0
0
1.000000
1
1000.000000
{Jammers Canberra - Hostile  - H523  - Alt Change}
tagin:=135; timein:=31.8917; squadin:=108; altitudein:=30000; ALT_UPDATE;
---
+++++
32.310001
3
0
0
1.000000
1
1000.000000
{Ground Attack MIG-27 BAHADUR - H0277 - Hostile - Alt Change}
tagin:=114; timein:=32.3033; squadin:=95; altitudein:=29000; ALT_UPDATE;
---
+++++
32.470001
3
0
0
1.000000
1
```

```
1000.000000
{Jammers Canberra - Hostile - H524  - Alt & Spd Change}
tagin:=136; timein:=32.47; squadin:=109; speedin:=580; altitudein:=31000; ALT_UPDATE; SPD_UPDATE;
---
+++++
32.529999
3
0
0
1.000000
1
1000.000000
{DCA F-16Cs - FALCON - FN21-26 - ID Change}
tagin:=20; timein:=32.53; squadin:=1044; ID:=211; ID_UPDATE;

---
+++++
32.720001
3
0
0
1.000000
1
1000.000000
{Ground Attack MIG-27 BAHADUR - H0301 - Hostile - Alt Change}
tagin:=116; timein:=32.72; squadin:=96; altitudein:=29000; ALT_UPDATE;
---
+++++
33.660000
3
0
0
1.000000
1
1000.000000
{Air Defense MIG-23 FLOGGER-B - H0431 - Hostile - Alt Change}
tagin:=58, timein:=33.6633; squadin:=67; speedin:=580; altitudein:=32000; ALT_UPDATE, SPD_UPDATE;
---
+++++
33.700001
3
0
0
1.000000
1
1000.000000
{DCA F-16Cs - SOLAR - SR11, 12, 13, 14, 15, 16 - Course Change}
tagin:=11, timein:=33.7; squadin:=1032; pattern:=0; n:=3; xin[1]:=-25; yin[1]:=240; xin[2]:=37; yin[2]:=125;  xin[3]:=23; yin[3]:=13;
NEW_COURSE;
---
+++++
33.820000
3
0
0
1.000000
1
1000.000000
{STRIKE F-16Ds  - HUSH 10 -  HH10-17, HH20-21 - Course Change}
tagin:=16, timein:=33.8167; squadin:=1077; pattern:=0; n:=3; xin[1]:=-25; yin[1]:=240; xin[2]:=37; yin[2]:=125;  xin[3]:=23; yin[3]:=13;
NEW_COURSE.
---
+++++
33.959999
3
0
0
1.000000
1
1000.000000
```

```
{Ground Attack JAGUAR - H0253 - Hostile}
tagin:=104; timein:=33.955; squadin:=90; squad_num:=2; speedin:=600; altitudein:=25000; pattern:=0; n:=5; xin[1]:=-110; yin[1]:=238;
xin[2]:=-100; yin[2]:=170; xin[3]:=0; yin[3]:=75; xin[4]:=-100; yin[4]:=170; xin[5]:=-110; yin[5]:=238; miss_ty_1:=0; miss_ty_2:=0; ID:=9999;
NEW_TRACK;
---
+++++
34.169998
3
0
0
1.000000
1
1000.000000
{DCA F-16Cs - SOLAR - SR17, 20, 21, 22 -  Course Change}
tagin:=145; timein:=34.1667; squadin:=1034; pattern:=0; n:=3; xin[1]:=-25; yin[1]:=240; xin[2]:=37; yin[2]:=125;  xin[3]:=23; yin[3]:=13;
NEW_COURSE;



---
+++++
34.250000
3
0
0
1.000000
1
1000.000000
{STRIKE F-16Ds  - HUSH 10 - HH22-27, HH30-33 - Course Change}
tagin:=143; timein:=34.25; squadin:=1078; pattern:=0; n:=3; xin[1]:=-25; yin[1]:=240; xin[2]:=37; yin[2]:=125;  xin[3]:=23; yin[3]:=13;
NEW_COURSE;
---
+++++
34.439999
3
0
0
1.000000
1
1000.000000
{DCA F-15E II - EAGLE 01 - EE13-17, EE20-24}
tagin:=146; timein:=34.435; squadin:=1064; squad_num:=10; speedin:=500; altitudein:=25000; pattern:=0; n:=7; xin[1]:=40; yin[1]:=75;
xin[2]:=0; yin[2]:=60; xin[3]:=-10; yin[3]:=-95; xin[4]:=-115; yin[4]:=211; xin[5]:=-25; yin[5]:=240; xin[6]:=-10; yin[6]:=145; xin[7]:=40;
yin[7]:=75; miss_ty_1:=0; miss_ty_2:=1; ID:=111; NEW_TRACK;
---
+++++
34.520000
3
0
0
1.000000
1
1000.000000
{MIG-21 Air Defense F-7M 1 - Hostile - H050- Alt Change}
tagin:=48; timein:=34.5233; squadin:=62; altitudein:=29000; ALT_UPDATE;
---
+++++
34.520000
3
0
0
1.000000
1
1000.000000
{Jammers Canberra - Hostile - H522 - Alt Change}
tagin:=134; timein:=34.5217; squadin:=107; altitudein:=28500; ALT_UPDATE;


---
+++++
35.709999
3
0
0
1.000000
1
1000.000000
{STRIKE F-111A  - LAZER 30 - LR36, 37, 40, 41}
tagin:=147; timein:=35.7133; squadin:=1089; squad_num:=4; speedin:=600; altitudein:=32000; pattern:=0; n:=5; xin[1]:=-10; yin[1]:=-95;
xin[2]:=-115; yin[2]:=211; xin[3]:=-25; yin[3]:=240; xin[4]:=-10; yin[4]:=145; xin[5]:=-5; yin[5]:=55; miss_ty_1:=0; miss_ty_2:=1; ID:=111;
NEW_TRACK;
---
+++++
35.959999
3
0
0
1.000000
1
1000.000000
{Ground Attack JAGUAR - H0253 - Hostile - ID Change}
tagin:=104; timein:=35.9567; squadin:=90; ID:=111; ID_UPDATE;
---
+++++
36.439999
3
0
0
1.000000
1
1000.000000
{DCA F-15E II - EAGLE 01 - EE13-17, EE20-24 - ID Change}
tagin:=146; timein:=36.4367; squadin:=1064; ID:=211; ID_UPDATE;
---
+++++
37.720001
3
0
0
1.000000
1
1000.000000
{STRIKE F-111A  - LAZER 30 - LR36, 37, 40, 41 - ID Change}
tagin:=147; timein:=37.715; squadin:=1089; ID:=211; ID_UPDATE;
---
+++++
37.959999
3
0
0
1.000000
1
1000.000000
{Ground Attack JAGUAR - H0253 - Hostile - ID Change}
tagin:=104; timein:=37.955; squadin:=90; ID:=121; ID_UPDATE;
---
+++++
38.330002
3
0
0
1.000000
1
1000.000000
{STRIKE F-15E  - ZIP 01 -  ZP01-07, ZP10-12 - Spd Change}
tagin:=15; timein:=38.333; squadin:=1072; speedin:=500; SPD_UPDATE;

---
+++++
39.959999
```

```
3
0
0
1.000000
1
1000.000000
{Ground Attack JAGUAR - H0253 - Hostile - ID Change}
tagin:=104; timein:=39.955; squadin:=90; ID:=141; ID_UPDATE;
---
+++++
44.000000
3
0
0
1.000000
1
1000.000000
halt();
---
+++++
********
2
601
100.000000
1
10.000000
1
200.000000
cie_data
cie_hvaa cie_fuel2 cie_arm cie_vector
---
+++++
1
260
100.000000
1
10.000000
1
200.000000
expert
com_time com_hvaa1 com_hvaa2 com_hostile  vector
---
+++++
2
500
100.000000
1
10.000000
1
200.000000
trace
clock task_num comp_1 comp_2 comp_3 com_hvaa1 com_hvaa2 com_hostile
---
+++++
********
```

Appendix B: APAD Assessor Description and Code

Introduction

This document describes the application programmer's interface of the APAD Bayesian belief network assessor module.

APAD Assessor's System Organization

The assessor is built into a DLL (Dynamic Link Library) using visual C++. The Assessor DLL supports building a student Bayesian network model and reasoning within that model. The assessor supports a "C" interface to LISP and other languages.

Input/Output Files in the Assessor

File Formats

Rules' Prior File. The Rules' prior file is used by the assessor to initialize the prior probabilities of the rules when a student uses the ITS for the first time. Rules' priors may be computed from observations of students, or they may be subjective estimates. All of the rule nodes are assumed to include two state values: F and T. The Rules prior file's EBNF representation is below.

```
Rule_file: rule_items
rule_items: rule_item
            rule_items rule_item
rule_item:  rule_id  rule_name rule_meaning 1- rule_prior_value rule_prior_value
rule_id:  integer
rule_name: identifier
rule_meaning: string
prior_value: double
```

After each rule_item, a new line is necessary here. All of the rules used in the assessor are put into "Rules.txt" as shown below.

Rules Prior File

```
1      CMonitor       Monitoring      0.5      0.5
2      CCategorization        Categorization        0.5      0.5
3      CROE ROE_Use       0.5      0.5
4      CSymbology   Symbology_knowledge        0.5      0.5
5      Cdata  Data_gathering      0.5      0.5
6      Csystem        System_use      0.5      0.5
```

```
7      Cdistance      Distance_calculation_from_data_or_screen_view   0.5      0.5
8      Ccomputation Computational_skills  0.5      0.5
9      Cvisualization Visual_comparison    0.5      0.5
10     CSA    Situational_awareness        0.5      0.5
11     Cplan  Planning      0.5      0.5
12     CHeuristic     Knowledge_of_heuristic        0.5      0.5
13     CAircraftSelection     Aircraft_selection    0.5      0.5
14     CDecisionmake          Decision_making      0.5      0.5
15     CAssets        Knowledge_of_assets 0.5      0.5
16     CVectoring     Knowledge_of_rules_for_HVAA_use_given_fuel_and_armament_status
       0.5      0.5
```

Old Rules' File. Old rules' file contains rules' old posterior probabilities of a student. It is created after the student solves the first problem and is updated after every problem. The old rule file should contain the same rules as the rules' prior file.

All of the rule nodes are assumed to include two state values: F and T. The old Rules prior file's EBNF is as same as Rules' prior file.

APAD Output File. The APAD comparison process output file is used as the evidences file. The format's EBNF is as follows.

```
Out_Put_File: Header Items
Header: String String String String String
Items: Item
       |Items tem
Item: Int Int Int Int String String
Int: Integer
```

An example of the APAD comparison output file is shown below.

APAD Comparison Output File

```
"cie_hvaa"    "cie_fuel2"    "cie_arm"     "cie_vector"  "Trigger"
1       1      3      1      "end 260"
1       1      3      1      "end 260"
3       1      3      1      "end 260"
1       1      3      1      "end 260"
3       1      3      1      "end 260"
1       1      3      1      "end 260"
3       1      3      1      "end 260"
1       1      3      3      "end 260"
1       1      3      3      "end 260"
3       1      3      2      "end 260"
3       1      3      2      "end 260"
1       1      1      3      "end 260"
```

3	*1*	*1*	*2*	*"end 260"*
3	*0*	*1*	*1*	*"end 260"*
2	*0*	*3*	*1*	*"end 260"*
2	*0*	*3*	*1*	*"end 260"*

APAD Bayesian Network. The Bayesian belief network represents APAD's reasoning model. It appears in Figure 3.

The Interface Functions of the APAD Assessor

The following code represents the interface functions of the assessor.

```
/**********************************************************************
 * Name:
 *   int Assessor_Load_Bayesian_Network(char *fname)
 * Function:
 *   Read the Bayesian network file into memory.
 * Results:
 *   Success: return 1
 *   Fail:   return other
 * Parameter:
 *   char *fname: the bayesian network file name.
 * Comment:
 *  (1)The file name should contain the entire path necessary for accessing the file.
 *  If it is not included the full path name, the default path is the current
 *  work directory.
 *  (2) The following format are supported:
 *   Microsoft format MSBN: format used by the program Microsoft Bayes Networks
 *   version 1.0.1.7 (API). Microsoft MSBN32 API library Copyright @ Microsoft
 *   Corp.
 *   Netica format:  format used by the program Netica@ version 1.06. Netica@ is a
 *   trade mark of Norsys Software Corp.
 *   Ergo format: format used by the program Ergo@ version 1.03. Ergo@ is a trade
 *   mark of Noetic Systems Incorporated.
 *   Hugin format : Format used by the program Hugin@ version 3.1.1 (API). Hugin is
 *   Copright @ Hugin Expert A/S.
 *   Dsl format: This is format for Genie.
**********************************************************************/
extern "C" _declspec(dllexport)
int Assessor_Load_Bayesian_Network(char *fname);

/************************************************************************
 * Name:
 *  void Assessor_Write_Bayesian_Network(char *fname)
* Function:
*   Write the bayesian network Bnet into a text file.
* Results:
*   void
* Parameter:
*  fname:  The name used to save the Bayesian network.
* Comments:
*  (1)The file name should contain the entire path necessary for accessing the file.
*  If it is not included the full path name, the default path is the current
*  work directory.
*  (2) The following format are supported:
*   Microsoft format MSBN: format used by the program Microsoft Bayes Networks
*   version 1.0.1.7 (API). Microsoft MSBN32 API library Copyright @ Microsoft
*   Corp.
*   Netica format:  format used by the program Netica@ version 1.06. Netica@ is a
*   trade mark of Norsys Software Corp.
*   Ergo format: format used by the program Ergo@ version 1.03. Ergo@ is a trade
*   mark of Noetic Systems Incorporated.
*   Hugin format : Format used by the program Hugin@ version 3.1.1 (API). Hugin is
*   Copright @ Hugin Expert A/S.
*   Dsl format: This is format for Genie.
*         This function writes the Bayesian network in Ergo format.
*Parameters:
*         char *fname: Bayesian network name.
*Results:
*         return 1: success
*         return other: fail
*Comments:
*   (1) This function is used to initial the Bayesian network.
*********************************************************************/
extern "C" _declspec(dllexport)
int Assessor_Write_Bayesian_Network(char *fname);

/*********************************************************************
*Name:
*  int Assessor_Read_Competency(char *fname)
*Function:
*         This function reads the competency rule's old information
*         and resets the rule node's prior by the post_probability value.
*Parameters:
*         char *fname: Competency file name
*Results:
*         return 1: success
*         return 0: fail
*Comments:
*         (1) Make sure the bnet/ rule nodes' name are fit into the id's
*  syntax.
*  (2) The rules file format is enclosed.
*     { integer_id       rule_idrule_name      prior_probabilities }*
*  (3) This file should be called after Assessor_Load_Bayesian_Network().
**********************************************************************/
extern "C" _declspec(dllexport)
int Assessor_Read_Competency(char *fname);

/**************************************************************
*Nmae:
*  double Assessor_Query_Node_Probability( char * node_id, int state)
*Function:
*        Query node's posterior_probability
*Parameters:
*        char * node_id : node's id
*        int state:  Node evidence state's position
*Results:
*        Success: double value (>> 0.0 and << 1.0).
*          It's the posterior-probability value of the node.
*  Fail:    -1. (node's id is wrong etc.)
*Comments:
*        (1)Make sure the bnet/ rule nodes' name are fit into the id's
*  syntax.
**************************************************************/
extern "C" _declspec(dllexport)
double Assessor_Query_Node_Probability( char * node_id, int state);

/*****************************************************************
*Name:
* int Assesssor_Read_Evidences_RollUp(char *file ,char *fname)
*Function:
* Read the evidences file, and updates the network, roll up the posterior
* probability of comptency rules after each evidences line till it's finished.
*Parameters:
* char *file: evidences file name. The file is Micro Saint model's output.
*      The file format is
*
* char *fname: used to save the posterior_probability of the compeltency rules after
*      the roll up.
*
*Results:
* return 1: success
* return 0: fail
*Comments:
* (1) This function should be called after the output file from Micro Saint
*  model is generated
```

```
*
* (&&&&&3) Here assume only one sequence evidences
*****************************************************************/
extern "C" _declspec(dllexport)
int Assesssor_Read_Evidences_RollUp(char *file ,char *fname);

/*****************************************************************
*Name:
*  int Assessor_Close_Problem(char *fname)
*Function:
*       (1)Close one problem and save the competency rules' posterior
*    probability into a file.
*  (2)Reset the Bayesian network to wait for the next problem.
*Parameters:
*  char * fname: file name used to save the post-probability value
*Results:
*       return 1: success
*       return other value: fail
*Comments:
*
*****************************************************************/
extern "C" _declspec(dllexport)
int Assessor_Close_Problem(char *fname);

/*****************************************************************
*Name:
*  int Assesssor_Finished(void )
*Function:
*       Assessor is finished, the student quit the system.
*       Free the competency file
*Parameters:
*       void
*Results:
*       return 1: success
*       return other: fail
*Comments:
*  This function should be called when the student quit the test.
*****************************************************************/
extern "C" _declspec(dllexport)
int Assesssor_Finished(void );
```

Assessor Application Trace File Example

This section demonstrates a "C" trace files for the assessor. Assessor.h is the interface file for the assessor DLL.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#ifndef ASSESSOR_H
#include "assessor.h"
#endif

void main(void)
{
//Load the MA&D bayesian network
Assessor_Load_Bayesian_Network("C:/MA_and_D/Assessor_test/MAADnet99.dsl");

//Read the rule's old posterior probability
Assessor_Read_Competency("C:/MA_and_D/Assessor_test/Rules.txt");

//Write the network into ergo format
Assessor_Write_Bayesian_Network("test.erg");

/*Read the output file from the Saint Model and save the posterior
 probabilites of rules into "test.old".
*/
Assesssor_Read_Evidences_RollUp("saint_output.txt","test.old");

//Close the problem
Assessor_Close_Problem(NULL);

//Quit the MA&D assessor system
Assesssor_Finished();

}
```

Appendix C: Review of Current Voice Recognition Capabilities for Support of AWACS Weapons Director Team Research

Background

In many high demand workplaces, teams of personnel coordinate their actions to accomplish a common set of goals. These teams may be doctors and nurses in an emergency room, a team of soldiers in a tank facing enemy fire, or they may occur in many other high stress situations and tense environments. For our military, one important team is comprised of the interacting personnel manning AWACS radar systems and the pilots who rely on them for guidance.

AWACS users form a team with the pilots they direct, and, as system users, they perform many complex tasks. Many of these tasks require extensive verbal communications with each other, with a primary team comprised of Weapons Directors (WDs), and aircraft pilots. The WDs interact with each other and the pilots to identify pilot information support needs. The WDs supply the pilots with tactical guidance, and position and course information concerning other aircraft, and schedule aircraft for rendezvous with fuel tankers. These communications consist of information transmittals necessary for other personnel to perform their own tasks. Thus, for full coordination of effort among WDs and pilots, effective and efficient communication is necessary.

To better aid coordinated teams or groups, such as WDs and pilots, in their training and performance, extensive research has been under way for many years to define, delineate, and assess individual and team behaviors; and to develop automated tools for supporting teams (see for example, Regian & Elliott, 1997; or Brannick, et al., 1997). Since verbal communication is such an integral aspect of the tasks performed by such coordinated teams, training and support tools, to best enhance team performance, should be able to capture, and include in analyses, the speech acts performed by team members, in addition to capturing other forms of observable behavior. The addition of this type of information would better inform the identification and analysis of task performance at both the team and individual level. The results of these analyses could then become the basis of automated job aiding or performance assessment and training tools.

In order to capture speech behavior, several approaches that can be taken. First, there is the time-consuming process of tape-recording speech, transcribing it manually, time-stamping it, and then including as a component of an analysis. Such an approach is not conducive to use in an automated system that would supply real-time input to the performer concerning his or her behavior. The ideal method would be to capture speech acts as they occur during an on-going situation, and as they happen, integrate them into a concurrent analysis of performance in order to give real-time feedback to the producer. This second method requires a data capture system that supports automatic speech recognition (ASR).

In the past, automated voice or speech recognition capabilities of data capture systems have been limited (Markowitz, 1996). These limitations were due to weaknesses in both the algorithms used to recognize specific speakers and parse speech and the power of the computer hardware to support the algorithms. Early systems frequently could support only very limited lexicons, and required many hours of training on a specific speaker's speech. Even after training, the speaker would have to talk very slowly and refrain from uttering words not contained in the lexicon. These early systems lacked the power necessary to allow for the integration of speech acts into automated analyses of team or group task performance.

Recently, however, great strides have occurred in ASR technology. Both the hardware to support speech recognition and the algorithms that are used in the process have increased significantly in power. The aim of this short report is to examine aspects of the current state of ASR research and available software to determine the potential for the use of this technology in the development of team training and job support aids, having a performance assessment component. The task group that will be used as the baseline for this assessment is that of AWACS weapons directors. To this end, this brief paper will:

- Review some of the basic goals of in automated speech recognition (ASR) programs,
- Define the requirements for ASR in support of teams such as WDs
- Present some of the more recent advances in ASR,
- Delineate the existing problems in ASR, with indications as to the progress toward their solutions, and
- Review the most promising of reasonably priced Commercial Off The Shelf (COTS) product to support the capture and analysis of Weapons Directors' speech acts.

This report will not delve into the mechanics of speech recognition, except in instances in which it is necessary to do so to clarify a point. The field of research is large in both depth and breadth, and this report will focus only on what is pertinent to determining how such research can meet the goals of team performance assessment, team training, and job support.

Goals for ASR

According to Markowitz (1996), "Speech is the ultimate, ubiquitous interface." Therefore it makes sense as a design goal, to develop a computer system with which humans can interact verbally. That is one of the primary aims of research in the area of ASR. To this end, researchers in ASR have focused on making the technology accessible and flexible. For example, there are products available, already, that support continuous speech or specialized vocabularies. There are systems incorporating ASR that require no training on individual speakers. There are ASR tool kits to allow developers to integrate ASR capabilities into products. However, as it will be shown, the goal of speech as the ubiquitous computer interface has not quite been reached.

Requirements for ASR Systems to Support Performance of Assessment of Teams

An ASR component of an automated performance assessment or data capture system for teams such as WDs and pilots has some very specific requirements. First, such a component should be able to recognize and differentiate multiple voices. In a situation in which several WDs are passing information to each other and to pilots, the ASR component will have to sort out the speech acts of all participants.

Second, the ASR component should support the recognition of a small, but specialized vocabulary. It is possible to pre-train the ASR component on the vocabulary prior to its use, however.

Third, the ASR component of such a data-gathering product would need to filter out a noisy environment. In the AWACS environment, there are extraneous noises, such as from the aircraft engines and from background chatter. Capturing these noises could lead to inaccurate collection of the target data.

Additionally, the ASR component should support continuous speech recognition, because of the nature of types of verbal tasks performed in team performance environments. An ASR component that requires the speakers to slow down from their normal rate of speech could lead to an inaccurate picture of both individual and team performance. Additionally, such a necessitated slow-down of speech could, potentially, lead to performance errors, which could be detrimental in a real-world situation.

Finally, an ASR product to be used as a component of an automated performance assessment or capturing tool would need to be capable of integration with other components of the tool. This means that it should be able to be accessed by other portion of the tool or access the other parts, as specified by the tool design. Part of this integration capability, should include the ability to allow for time calibration and time stamping of the individual speech acts in relation to the on-going events to which the team is responding.

Recent Advances

In some respects, ASR is still a new technology. However, great strides have taken place in the field in the last eight years. Many of these advances are important when considering the use of ASR as part of a team performance assessment or data capture tool. Some of these recent advances are described below.

One major requirement of a team performance tool is the ability to recognize verbalizations by multiple speakers. Krasinski and Sukkar at AT&T (1994) have developed a telephone menu system that recognizes multiple speakers. The system also does not require training for recognition of users speaking words within the system's vocabulary. The system also "listens" for speech acts that occur while an automated voice is supplying directions to the system users. Finally, the system is designed to ignore speech acts that are not supported by its available vocabulary. It has a rate of 97.8% successful recognition of speech acts presented to it.

Since the fielding of this application, several telephone companies have fielded similar, very powerful automated menu systems that use ASR.

Another research effort, by Klevens (1997), has focused on a combination of problems that are all pertinent to the team performance situation. In his work, he has developed an ASR system to capture and translate into text dialogue between police detectives and interviewees. In this type of event, multiple voices need to be captured and sorted. Additionally, there is no training period for the software with regard to the interviewees, nor is there a way to have a stabilized vocabulary available to the ASR system. Finally, Klevens was able to build in two types of filters, one to decrease the negative impact of background noises to speech recognition and a second one to reduce the distortions to human voices that are produced by sound transmittal through microphones. The result of this effort was a 99.6% accuracy rate for recognition of the performed speech acts. However, in his conclusions, Klevens felt that more work was needed in the area of sound filtering, so that the accuracy rate of recognition could be increased.

As mentioned in the previous section, for ASR to be of use in the assessment of teams that use speech acts as part of their task performance, the ability of the system to recognize continuous speech is imperative. Recent advances in ASR technology have allowed for the commercialization of products, such as products by Dragon Systems, Phillips, and IBM, that can recognize up to 160 words per minute, without pauses between words. In the product reviewed below, one such ASR system is examined extensively.

The commercial products that have appeared recently on the market also promise support for extensive and easily modified vocabularies. For teams such as WDs and pilots, the capability to recognize a large vocabulary, in itself, is not important, since the terms allowed in the situation are limited and prescribed. However, the capability for easy vocabulary customization is important, given the specificity of the terminology used by WDs and pilots.

Finally, to be useful as an integrated component of a larger automated tool, an ASR product needs to be available in an form that can be called by other components of the tool, as needed. There are several companies involved in the development of ASR tools that offer such integratable ASR products (engines) in the form of developers' toolkits. One such package is ART's SmARTpeak SDK (Software Developer's Kit) that offers an Application Program Interface (API) that allows ART's ASR engine to be integrated with other applications developed in programming languages such as C or Visual Basic. A similar package is offered by Voice Control Systems. These packages seem to offer a means to begin the development of a very useful automated tool that needs to capture voice information to be input into its other functions.

Problems

Although there have been extensive advances in the area of ASR, many of these changes appear to be due to the increase in power of the personal computer systems used to run the applications. With the increase of processing speed and power, has come the support for the complex algorithms ASR products use. The increase in storage space available on current PCs can support the extensive vocabularies. There remain several problems with ASR systems, however. ASR systems, for the most part, still require close-talking microphones, a single speaker, and a relatively quiet background for optimal performance.

Additionally, researchers such as Dr. Victor Zue of MIT (1985) have indicated that the types of domain- and user-independent statistical sampling algorithms, such as Hidden Markov Models, used in most ASR products may not be powerful enough to allow the development of products that will overcome current shortcomings. The methodology that has been pursued for over twenty years is to develop domain-independent applications, and according to Dr. Zue, as early as 1985, this approach may have reached its limits with regard to extendibility. The successes in products that use such an approach are due in large part to increases in power of memory availability of the supporting systems. Some researchers, including Dr. Zue (1995), are investigating the development of ASR systems that use domain-specific rules. Unfortunately, this new approach to the technology is not moving into the marketplace very quickly. This approach seems to be potentially useful for the solution of the problems of environmental noise filtration and support of multiple speakers. These issues are discussed below.

There have been some very exciting research trends using domain-specific rules in the area of environmental noise accommodation for ASR systems. For example, Nawab, et al., (1998) have been pursuing the approach of training ASR systems to recognize domain-specific noises, so that the application can sort the background sounds out from the human speech acts. Additionally, Klevens' work mentioned above includes the design of special filters that allow the ASR system to "ignore" de-humanizing aspects of transmitted sound and common background noises.

Similarly, Defense Group Incorporated (DGI), has been focusing specifically on the ASR-impacting background noise interference found frequently in military environments. Their approach is to remove the background noise via adaptive noise cancellation processes and adaptive noise power estimation algorithms coupled with nonlinear noise pre-filtering prior to subsequent processing. This pre-filtering process is then combined with a secondary filtering process, which uses a neural network trained on the background noises common to the environment. These noises are also removed prior to further speech processing. More information concerning this effort may be found on the Internet at http://www.ca.defgrp.com/noise.html.

Finally, researchers such as Klevens, and Krasinski and Sukkar, above, are investigating the issue of support for multiple speakers. One approach is to pre-train the ASR system to recognize multiple speakers. That is the approach taken by current off-the-shelf products. Other approaches, implemented in some break-through packages used for conference transcription, use rule-based algorithms rather than stochastically based approaches to drive the recognition process. As mentioned, some researchers, such as Zue (1995), have indicated that the use of rule-based systems or neural net algorithms that better model the human auditory process will allow future ASR systems to overcome many of their current drawbacks. Such work is currently in progress at Massachusetts Institute of Technology's Laboratory for Computer Science, the

Automatic Speech Recognition Laboratory at the University of Illinois, and at *Fonix* Corporation.

Ability of COTS ASR Products to Support Team Performance Assessment and Job Support

A search of the popular literature in the computer field was performed to select an off-the-shelf product for evaluation to determine its ability to capture voice data and to supply the functionality necessary to meet the requirements of automated team performance assessment . It was decided that rather than assessing the performance of multiple products, only the product that had received the most critical recognition and promised the most advanced features would be examined. It was felt that the selection would best represent the state-of-the-art for commercially available products, keeping in mind that continuing research efforts promise even more-greater enhanced products in the future.

Dragon NaturallySpeaking Preferred Version 3 by Dragon Systems was selected for evaluation. This product was selected because of the following advertised features:

- Support for continuous speech
- 95% or higher accuracy rate
- Support for automatically specializing the recognized vocabulary
- Support for multiple speakers, with short requirements for the program to learn to identify the individuals
- Ability to integrate with other off-the-shelf products such as Word for Windows

Each of these features was seen as important in the ability of such a product to support the identification of verbal task performance by some automated assessment tool. The importance of each feature is described in the following paragraphs.

Support for Continuous Speech—When WDs perform their tasks, they passing information along in a natural, continuous manner. They cannot be inconvenienced by an ASR product that requires them to slow down so that the product can capture their speech. Such a slow-down could be fatally detrimental to their job performance.

High Accuracy Rate—To be at all useful in performance assessment, either in support of training or job performance, an ASR tool must capture data accurately. Inaccurate data could lead to other portions of the assessment or training tool to come up with incorrect results and guidance.

Support for Automatically Specializing the Recognized Vocabulary—Weapons Directors and pilots use a unique vocabulary. It is not large, however. Any ASR tool that is used for data capture must, however, be able to recognize the unique items that used in WD and pilot task performance.

Support for Multiple Speakers—During a mission or situation in which a team is acting, there will be more than one individual speaking. Therefore to be of any use in the assessment of team performance, and ASR data capture tool must be able to differentiate between speakers. Although NaturallySpeaking does not have the capability to perform this task in that individual contributions cannot be tagged, it can recognize multiple speakers. This indicates that a future product could easily be developed so that verbal performance by multiple speakers within the same situation could be captured and sorted.

Integration Ability—NaturallySpeaking allows for direct input into easily available word processing products, such as Word for Windows. It is not a large step to go from input into word processing software into input, say as an ASII file, with utterances time-stamped, for another application, such as a performance assessment tool.

Product Assessment

The selected product was assessed with regard to the following factors:

- Training requirements
- Multiple speaker recognition
- Ease of expanding the speech variation and vocabulary
- Impact of background noise

Training Requirements. The initial training of the product was straightforward, although it required approximately 30 minutes to do so. The training consisted of the author of this report reading a selection of provided text. A subsequent test, in which the user began free dictation, revealed the need for additional training to recognize changes in the user's speech pattern resulting from the use of vocabulary words not previously trained.

An additional training session of 30 minutes was required to reach a level of 98% accuracy for one user. This training session included adding vocabulary from documents that were analyzed by the product to identify new words. Also, a subsequent session in which the user spoke random sentences and phrases (some of which would be similar to those spoken by Weapons Directors) and trained the product on specific unrecognized phrases. The user then read a portion of a textbook on Automatic Speech Recognition to the product to determine its ability to recognize the words in the passage. It was during this last test about the product reached the level of 98% accuracy. It should be noted that the product allows one to set a speed-accuracy trade-off level that is acceptable to the individual. The 98% level was reached with the product set with the speed-accuracy trade-off placed at a middle level.

Multiple Speaker Recognition. NaturallySpeaking allows for the development of multiple speaker files. The system is designed such that at the beginning of its use, one must specify the speaker. Then the software accesses that user's speech and vocabulary files to allow for ASR to proceed. Currently, the product does not support multiple speakers performing within a single session.

Ease of Expanding Recognition of Speech Variation and Vocabulary. There are two methods for expanding the speech variation vocabulary for an individual speaker: 1) dictate a document, at the completion of which, the tool will ask if the speech files should be updated, and 2) processing specified documents that reflect the user's standard vocabulary or writing style, with a request for the user to verbally input the words found in those documents that have not been previously encountered during training or use.

Impact of Background Noise. A brief evaluation of the product's ability to recognize a user's speech acts with a loud background sound was performed. Please note that this was not a rigorous test, but one to supply information as to the potential utility of the product for prototype performance assessment tools for team behaviors.

Once the product was trained to recognize a user at an accuracy rate of 98% over multiple paragraphs of dictation under the background noise conditions found in a quiet office, a loud, continuous background noise was introduced This background noise produced by a running large, six-horsepower industrial vacuum cleaner placed three feet from the user during dictation. We felt that such a loud noise would allow some conclusions to be made about the robustness of the product with regard to environmental sound. We are well aware that sound from such a vacuum cleaner may not have all the properties of the background environment noises found in an AWACS aircraft. However, we felt that if the vacuum cleaner noise interfered with ASR by the product, then it probably would not perform acceptably within an environment with potentially greater decibels of sound.

The results of this small evaluation were that the product's performance degraded from a high of 98% accuracy on recognition of the contents of a five paragraph reading under quiet, office conditions to 93% on the same reading under extremely noisy conditions. However, training the product on individual words under extremely noisy conditions probably could boost the accuracy. Additionally, reducing the speed of recognition by the product would also increase accuracy. It should be pointed out that the product is shipped with a sensitive microphone and an attached device (Parrot Translator) to help reduce the impact of environmental noise. This device was used at all times with the product.

To give a final example of the strength of the product, this final paragraph was dictated into Word 97 for Windows™ using the product. Within the first sentence one error was found. The error was rapidly corrected, using the available support tools. As a support for rapid dictation, this product seems to be adequate. For example, this paragraph was dictated at the speed of normal speech. However, to support the needs of Weapons Directors during either training or actual on the job performance, some enhancements would be required. The concluding section of this paper summarizes these requirements.

Conclusions

In this paper, a brief review of the most recent findings concerning ASR technology as it pertains to team performance data capture and assessment has been presented. Additionally, a rudimentary investigation of capabilities of the most highly reviewed and well-known COTS ASR product was also performed. From these two investigations, our conclusion is that much of the ASR technology is close to the state where it can be integrated into team performance and communications assessment tools. The important word here is "integrated." There is ASR technology available that will meet many of the requirements for use in a tool such as the one envisioned in this paper. However, work must be undertaken to bring these pieces together.

For example, the state of the technology is such that multiple speaker recognition is currently available in conference transcription software, and on-going research shows great promise for this technology. Additionally, the work in environmental noise removal, or reduction, seems to be moving swiftly such that the capability should be available in an ASR tool in the very near future. ASR technology already supports continuous speech recognition and relatively easy methods for training voice files, with profiles being updated after every session with the user. Finally, there are programmer tool kits to allow ASR technology to be integrated into other applications. (Possibly, such an application kit would allow for the integration of ASR technology with a time-stamping capability needed to synchronize other captured performance data with the verbal acts.) These pieces need to be brought together in a single package before one can begin to integrate it with other components for comprehensive automated performance assessment, job aiding, or training tools.

References

Brannick, M. T., Salas, E., & Prince, C. (1997). *Team performance assessment and measurement : Theory, methods, and applications (Series in Applied Psychology).* Mahwah, NJ: Lawrence Erlbaum Associates.

Klevens, R.L. (1997). *Voice recognition*. Boston, MA: Artech.

Krasinski, D. J. & Sukkar, R. A. (1994). Automatic speech recognition technology for network call routing. In *IVTTA: IEEE workshop on interactive voice technology for telecommunications applications*, September 1994, Kyoto, Japan.

Markowitz, J. A. (1996). *Using speech recognition.* Upper Saddle River, NJ: Prentice-Hall PTR.

Nawab, S. H., Espy-Wilson, C. Y., Mani, R., & Bitar, N. N. (1998). Knowledge-based analysis of speech mixed with sporadic environmental sounds. In Rosenthal, D. F. & Okuno, H.G. (Eds.) *Computational auditory scene analysis*. 177-214. Mahwah, NJ: Lawrence Erlbaum Associates.

Regian, J. W. & Elliott, L. R. (1997). *Intelligent tutoring for team performance: Theory-based development of an enhanced platform for team training and performance research.* Technical paper in review, Armstrong Laboratory, Brooks AFB, TX.

Zue, V. W. (1985). The use of speech knowledge in automatic speech recognition. In *Proceedings of the IEEE 73*, No. 11, November 1985, p. 1602.

Zue V. W. (1995). *Research overview, 1995 annual research summary.* Laboratory for Computer Science, Massachusetts Institute of Technology, Boston, MA: MIT.

Appendix D: Project Participants

The project participants for this effort from Micro Analysis & Design, Inc., were:

Dr. Debra Evans, Principal Investigator

Mr. John Keller, MicroSaint Modeler

Ms. Shelly Scott-Nash, Programmer

The project participants from the University of Pittsburgh were:

Dr. Kurt VanLehn, University Research Partner

Mr. Zhendong Nie, Programmer

The project participants from Veridian Corporation were:

Mr. Mathieu Dalrymple, Domain Expert

Mr. Phillip Tessier, Programmer Analyst